z/OS

MVS Diagnosis: Tools and Service Aids

GA22-7589-03

z/OS

MVS Diagnosis: Tools and Service Aids

GA22-7589-03

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page B-1.

Fourth Edition, September 2002

This is a major revision of GA22-7589-02.

This edition applies to Version 1 Release 4 of z/OS™ (5694-A01), Version 1 Release 4 of z/OS.e (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order documents through your IBM® representative or the IBM branch office serving your locality. Documents are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures . . . xiii

Tables . . . xv

About this document . . . xvii
Who should use this document. . . . xvii
Where to find more information . . . xvii
Information updates on the web . . . xviii
Accessing z/OS licensed documents on the Internet. . . . xviii
Using LookAt to look up message explanations . . . xviii

Summary of changes . . . xxi

Chapter 1. Selecting Tools and Service Aids . . . 1-1
How Do I Know Which Tool or Service Aid to Select? . . . 1-1
What Tools and Service Aids are Available? . . . 1-3
Dumps . . . 1-3
Traces . . . 1-4
Service Aids . . . 1-5

Chapter 2. SVC Dump . . . 2-1
Planning Data Set Management for SVC Dumps . . . 2-2
Using Automatically Allocated Dump Data Sets . . . 2-2
Using Pre-Allocated Dump Data Sets . . . 2-7
Using Extended Sequential Data Sets . . . 2-8
Obtaining SVC Dumps . . . 2-11
Issuing a Macro for SVC Dump . . . 2-12
Operator Activities . . . 2-12
Making a Dump Data Set Available. . . . 2-15
Determining Current SVC Dump Options and Status . . . 2-16
Finding SVC Dumps . . . 2-17
Printing, Viewing, Copying, and Clearing a Pre-Allocated or SYS1.DUMPxx Data Set. . . . 2-20
Contents of SVC Dumps . . . 2-21
Customizing SVC Dump Contents . . . 2-21
Tailoring SVC Dumps. . . . 2-31
Analyzing Summary SVC Dumps . . . 2-32
SUMDUMP Output For SVC-Entry SDUMPX . . . 2-33
SUMDUMP Output for Branch-Entry SDUMPX . . . 2-34
Analyzing Disabled Summary Dumps . . . 2-34
Analyzing Suspend Summary Dumps . . . 2-35
Analyzing an SVC Dump . . . 2-36
Specifying the Source of the Dump. . . . 2-36
Formatting the SVC Dump Header . . . 2-37
Looking at the Dump Title . . . 2-38
Displaying the Incident Token, Time and Type of Dump . . . 2-39
Locating Error Information . . . 2-40
Analyze TCB Structure . . . 2-43
Examining the LOGREC Buffer . . . 2-44
Examining the System Trace . . . 2-47
Looking at the Registers. . . . 2-47
Other Useful Reports for SVC Dump Analysis . . . 2-49
Reading the SDUMPX 4K SQA Buffer . . . 2-50

Chapter 3. Transaction Dump 3-1
Planning Data Sets for Transaction Dumps 3-2
Using Pre-Allocated Dump Data Sets 3-2
Using Automatically Allocated Dump Data Sets 3-3
Obtaining Transaction Dumps 3-5
Issuing a Macro for Transaction dump 3-6
Printing, Viewing, Copying, and Clearing a Pre-Allocated or a Dump Data Set 3-6
Contents of Transaction Dumps 3-7
Customizing Transaction Dump Contents 3-7
Analyzing Summary Transaction Dumps 3-13
SUMDUMP Output for IEATDUMP 3-14
Analyzing a Transaction Dump 3-14
Specifying the Source of the Dump. 3-15
Formatting the Transaction Dump Header 3-15
Looking at the Dump Title 3-15
Displaying the Time and Type of Dump 3-17
Locating Error Information 3-17
Analyze TCB Structure 3-20
Examining the LOGREC Buffer 3-22
Examining the System Trace 3-25
Looking at the Registers. 3-25
Other Useful Reports for Transaction Dump Analysis 3-26

Chapter 4. Stand-Alone Dump 4-1
Planning for Stand-Alone Dump 4-2
Should I take a stand-alone dump to DASD or to tape?. 4-2
Can I use my current version of the stand-alone dump program to dump a new version of MVS? 4-6
Creating the Stand-Alone Dump Program 4-6
MNOTES from the AMDSADMP Macro. 4-6
Coding the AMDSADMP Macro 4-11
Generating the Stand-Alone Dump Program 4-28
Two-Stage Generation 4-31
Running the Stand-Alone Dump Program 4-36
Procedure A: Initialize and Run Stand-Alone Dump 4-37
Procedure B: Restart Stand-Alone Dump 4-40
Procedure C: ReIPL Stand-Alone Dump 4-41
Procedure D: Dump the Stand-Alone Dump Program 4-41
Running the Stand-Alone Dump Program in a Sysplex 4-41
Capturing a Stand-Alone Dump Quickly 4-43
Minimize the Operator Actions 4-43
Get a Partial Stand-Alone Dump. 4-44
Copying, Viewing, and Printing Stand-Alone Dump Output 4-44
Copying the Dump to a Data Set 4-44
Viewing Stand-Alone Dump Output. 4-47
Printing Stand-Alone Dump Output 4-48
Message Output 4-49
Stand-Alone Dump Messages on the 3480, 3490, or 3590 Display 4-49
Analyzing Stand-Alone Dump Output 4-50
Collecting Initial Data 4-50
Analyzing an Enabled Wait. 4-53
Analyzing a Disabled Wait 4-57
Analyzing an Enabled Loop 4-58
Analyzing a Disabled Loop. 4-58
SLIP Problem Data in the SLIP Work Area 4-60
Problem Data Saved by First Level Interrupt Handlers. 4-60

Chapter 5. ABEND Dumps . . . 5-1
Synopsis of ABEND Dumps . . . 5-1
Obtaining ABEND Dumps. . . . 5-3
Data Set for Dump . . . 5-3
Process for Obtaining ABEND Dumps . . . 5-5
Printing and Viewing Dumps. . . . 5-7
Contents of ABEND Dumps . . . 5-8
Determining Current ABEND Dump Options . . . 5-9
Default Contents of Summary Dumps in ABEND Dumps . . . 5-13
Customizing ABEND Dump Contents . . . 5-14
Customizing SYSABEND Dump Contents . . . 5-16
Customizing SYSMDUMP Dump Contents . . . 5-17
Customizing SYSUDUMP Dump Contents . . . 5-19
Analyzing an ABEND Dump . . . 5-20
Analysis Procedure . . . 5-21

Chapter 6. SNAP Dumps . . . 6-1
Obtaining SNAP Dumps . . . 6-1
Customizing SNAP Dump Contents . . . 6-4
Customizing through Installation Exits . . . 6-4
Customizing through the SNAP or SNAPX Macro . . . 6-5

Chapter 7. The Dump Grab Bag. . . . 7-1
Problem Data for Storage Overlays . . . 7-1
Analyzing the Damaged Area . . . 7-1
Common Bad Addresses . . . 7-2
Problem Data from the Linkage Stack . . . 7-3
Problem Data for Modules . . . 7-4
Processing Modes . . . 7-4
Problem Data from Recovery Work Areas. . . . 7-4
Problem Data for ACR . . . 7-5
Pre-Processing Phase Data . . . 7-5
Data Obtained by IPCS . . . 7-5
Problem Data for Machine Checks . . . 7-6

Chapter 8. System Trace . . . 8-1
Customizing System Tracing . . . 8-1
Increasing the Size of the System Trace Table . . . 8-1
Tracing Branch Instructions . . . 8-2
Receiving System Trace Data in a Dump . . . 8-2
Formatting System Trace Data in a Dump. . . . 8-3
Reading System Trace Output . . . 8-3
Example of a System Trace in a Dump. . . . 8-3
Summary of System Trace Entry Identifiers . . . 8-4
ACR Trace Entries . . . 8-6
ALTR Trace Entries . . . 8-7
BR Trace Entries . . . 8-8
BSG, PC, PR, PT, and SSAR Trace Entries . . . 8-9
MODE and MOBR Trace Entries. . . . 8-11
DSP, SRB, SSRB, and WAIT Trace Entries. . . . 8-12
CSCH, HSCH, MSCH, RSCH, SSCH and SIGA Trace Entries. . . . 8-13
CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries . . . 8-15
SUSP Trace Entries . . . 8-17
PGM and SPER Trace Entries . . . 8-19
RCVY Trace Entries . . . 8-20
SVC, SVCE, and SVCR Trace Entries . . . 8-24

SSRV Trace Entries 8-25
TIME Trace Entries 8-28
USRn Trace Entries 8-29

Chapter 9. Master Trace 9-1
Master Trace and the Hardcopy Log 9-1
Customizing Master Trace 9-2
Requesting Master Trace 9-2
Receiving Master Trace 9-3
Reading Master Trace Data 9-4
Master Trace Output Formatted in a Dump 9-4
Master Trace Table in Storage 9-5

Chapter 10. The Generalized Trace Facility (GTF) 10-1
GTF and IPCS 10-2
GTF and the GTRACE Macro. 10-2
GTF and Indexed VTOC Processing 10-2
Using IBM Defaults for GTF 10-3
The IBM-Supplied Parmlib Member of GTF Trace Options 10-3
The IBM-Supplied Cataloged Procedure 10-3
Customizing GTF 10-4
Defining GTF Trace Options 10-4
Setting Up a Cataloged Procedure 10-4
Determining GTF's Storage Requirements 10-9
Starting GTF 10-10
Using the START Command to Invoke GTF 10-11
Specifying or Changing GTF Trace Options through System Prompting 10-12
Examples of Starting GTF 10-13
Starting GTF to Trace VTAM Remote Network Activity 10-16
Stopping GTF 10-16
Example of Stopping GTF 10-18
GTF Trace Options 10-18
Combining GTF Options 10-23
Examples of Sample Prompting Sequences 10-29
Receiving GTF Traces 10-31
Combining, Extracting, and Merging GTF Trace Output 10-32
Merging Trace Output 10-33
Reading GTF Output 10-34
Formatted GTF Trace Output 10-35
Trace Record Identifiers 10-36
Example Formatted GTF Trace output 10-37
Formatted Trace Records for Events. 10-39
Time Stamp Records 10-39
Source Index Records 10-40
Lost Event Records 10-40
CCW Trace Records 10-41
CSCH and HSCH Trace Records 10-42
DSP and SDSP Trace Records. 10-44
EOS, CS, IO, and PCI Trace Records 10-45
EXT Trace Records 10-47
FRR Trace Records 10-49
HEXFORMAT, SUBSYS, and SYSTEM Trace Records 10-50
IOX Trace Records 10-51
LSR Trace Records 10-54
MSCH Trace Records 10-55
PGM and PI Trace Records 10-56

RNIO Trace Records . . . 10-57
RSCH Trace Records . . . 10-58
SLIP Trace Records . . . 10-59
SLIP Standard Trace Record . . . 10-59
SLIP Standard/User Trace Record . . . 10-62
SLIP User Trace Record . . . 10-63
SLIP Debug Trace Record . . . 10-63
SRB Trace Records . . . 10-64
SRM Trace Records . . . 10-65
SSCH Trace Records . . . 10-66
STAE Trace Records . . . 10-67
SVC Trace Records . . . 10-69
USR Trace Records . . . 10-70
Unformatted USR Trace Record . . . 10-71
Formatted USR Trace Record . . . 10-71
USRF9 Trace Record for VSAM . . . 10-72
USRFD Trace Record for VTAM . . . 10-73
USRFE Trace Record for BSAM, QSAM, BPAM, and BDAM . . . 10-73
USRFF Trace Record for Open/Close/EOV Abnormal End . . . 10-74
USRFF Trace Record for User-Requested Work Area . . . 10-74
Event Identifiers (EIDs) for USR Trace Records . . . 10-75
Format Identifiers (FIDs) for USR Trace Records . . . 10-76
Unformatted GTF Trace Output . . . 10-77
Control Records . . . 10-77
Unformatted Lost Event Records . . . 10-79
User Data Records . . . 10-80
System Data Records . . . 10-81
Unformatted Trace Records for Events . . . 10-82

Chapter 11. Component Trace . . . 11-1
Planning for Component Tracing . . . 11-3
Create CTncccxx Parmlib Members for Some Components . . . 11-3
Select the Trace Options for the Component Trace . . . 11-8
Decide Where to Collect the Trace Records . . . 11-9
Obtaining a Component Trace . . . 11-10
Request Component Tracing to Address-Space or Data-Space Trace Buffers . . . 11-10
Request Writing Component Trace Data to Trace Data Sets . . . 11-13
Create a Parmlib Member . . . 11-16
Request Component Tracing for Systems in a Sysplex . . . 11-18
Verifying Component Tracing . . . 11-21
Verify that the Writer Is Active . . . 11-23
Viewing the Component Trace Data . . . 11-23
SYSAPPC Component Trace . . . 11-25
Requesting a SYSAPPC Trace . . . 11-26
Formatting a SYSAPPC Trace . . . 11-29
Output from a SYSAPPC Trace . . . 11-33
FMH-5 Trace Data . . . 11-35
SYSDLF Component Trace . . . 11-39
Requesting a SYSDLF Trace . . . 11-39
Formatting a SYSDLF Trace . . . 11-39
Output from a SYSDLF Trace . . . 11-40
SYSDSOM Component Trace . . . 11-41
Requesting a SYSDSOM Trace . . . 11-41
Formatting a SYSDSOM Trace . . . 11-41
Output from a SYSDSOM Trace . . . 11-42

SYSGRS Component Trace 11-43
Requesting a SYSGRS Trace 11-44
Formatting a SYSGRS Trace 11-47
Output from a SYSGRS Trace 11-48
CTRACE COMP(SYSGRS) TALLY Subcommand Output 11-48
SYSIEFAL Component Trace 11-49
Requesting a SYSIEFAL Trace 11-49
Formatting a SYSIEFAL Trace 11-52
Output from a SYSIEFAL Trace. 11-53
CTRACE COMP(SYSIEFAL) FULL Subcommand Output 11-53
SYSIOS Component Trace 11-54
Requesting a SYSIOS Trace. 11-55
Formatting a SYSIOS Trace 11-58
CTRACE COMP(SYSIOS) Subcommand Output 11-58
SYSJES Component Trace 11-60
Requesting a SYSJES Trace 11-62
Formatting a SYSJES Trace 11-65
Output from a SYSJES Trace 11-66
SYSjes2 Component Trace 11-71
Requesting a SYSjes2 Trace 11-71
Formatting SYSjes2 Sublevel Trace Information. 11-71
Output from a SYSjes2 Trace 11-72
SYSLLA Component Trace 11-73
Requesting a SYSLLA Trace. 11-74
Formatting a SYSLLA Trace 11-74
SYSLOGR Component Trace 11-74
Getting a Dump of System Logger Information 11-75
Requesting a SYSLOGR Trace. 11-77
Formatting a SYSLOGR Trace 11-80
Output from a SYSLOGR Trace 11-81
SYSOMVS Component Trace 11-81
Requesting a SYSOMVS Trace. 11-82
Formatting a SYSOMVS Trace 11-84
Output from a SYSOMVS Trace 11-86
CTRACE COMP(SYSOMVS) SUMMARY Subcommand Output 11-91
Output from a SYSOMVS Trace Using the SCCOUNTS Option 11-91
SYSOPS Component Trace 11-92
Requesting a SYSOPS Trace 11-93
Formatting a SYSOPS Trace 11-95
Output from a SYSOPS Trace 11-96
CTRACE COMP(SYSOPS) FULL Subcommand Output. 11-96
SYSRRS Component Trace 11-97
Requesting a SYSRRS Trace 11-98
Formatting a SYSRRS Trace 11-101
Output from a SYSRRS Trace. 11-102
SYSRSM Component Trace 11-105
Requesting a SYSRSM Trace 11-106
Formatting a SYSRSM Trace 11-119
Output from a SYSRSM Trace 11-119
SYSSPI Component Trace 11-120
Requesting a SYSSPI Trace 11-121
Formatting a SYSSPI Trace 11-121
SYSTTRC Transaction Trace 11-121
SYSVLF Component Trace. 11-121
Requesting a SYSVLF Trace 11-122
Formatting a SYSVLF Trace 11-123

Output from a SYSVLF Trace . . . 11-123
SYSWLM Component Trace . . . 11-125
Requesting a SYSWLM Trace . . . 11-125
Formatting a SYSWLM Trace . . . 11-126
Output from a SYSWLM Trace . . . 11-126
CTRACE COMP(SYSWLM) FULL Subcommand Output . . . 11-126
SYSXCF Component Trace . . . 11-127
Requesting a SYSXCF Trace . . . 11-128
Formatting a SYSXCF Trace . . . 11-130
Output from a SYSXCF Trace . . . 11-131
SYSXES Component Trace . . . 11-131
Requesting a SYSXES Trace . . . 11-133
Formatting a SYSXES Trace . . . 11-135
Output from a SYSXES Trace . . . 11-136

Chapter 12. Transaction Trace . . . 12-1
How Transaction Trace Works . . . 12-1
Transaction Trace Commands . . . 12-2
The TRACE TT Command . . . 12-2
DISPLAY TRACE,TT . . . 12-4
Using IPCS To View Transaction Trace Output . . . 12-4
IPCS CTRACE COMP(SYSTTRC) Examples . . . 12-4

Chapter 13. GETMAIN, FREEMAIN, STORAGE (GFS) Trace . . . 13-1
Starting and Stopping GFS Trace . . . 13-1
Receiving GFS Trace Data . . . 13-3
Formatted GFS Trace Output . . . 13-4
Unformatted GFS Trace Output . . . 13-5

Chapter 14. Recording Logrec Error Records . . . 14-1
Collection of Software and Hardware Information . . . 14-1
Choosing the Correct Logrec Recording Medium . . . 14-2
Initializing and Reinitializing the Logrec Data Set . . . 14-2
Initializing the Logrec Data Set . . . 14-3
Reinitializing the Logrec Data Set . . . 14-4
Defining a Logrec Log Stream . . . 14-4
Error Recording Contents . . . 14-6
Logrec Data Set Header Record . . . 14-7
Logrec Data Set Time Stamp Record . . . 14-8
Types of Logrec Error Records . . . 14-8
Obtaining Information from the Logrec Data Set . . . 14-10
Using EREP . . . 14-10
Obtaining Records from the Logrec Log Stream . . . 14-12
Using System Logger Services to Obtain Records from the Logrec Log Stream . . . 14-12
Using EREP to Obtain Records from the Logrec Log Stream . . . 14-12
Obtaining Information from the Logrec Recording Control Buffer . . . 14-18
Formatting the Logrec Buffer . . . 14-18
Finding the Logrec and WTO Recording Control Buffers . . . 14-18
Reading the Logrec Recording Control Buffer . . . 14-19
Interpreting Software Records . . . 14-19
Detail Edit Report for a Software Record . . . 14-20

Chapter 15. AMBLIST . . . 15-1
Obtaining AMBLIST Output . . . 15-1
Specifying the JCL Statements . . . 15-2

Controlling AMBLIST Processing 15-2
Examples of Running AMBLIST 15-6
List the Contents of an Object Module 15-6
Map the CSECTs in a Load Module or Program Object 15-8
Trace Modifications to the Executable Code in a CSECT 15-11
List the Modules in the Link Pack Area and the Contents of the DAT-On Nucleus 15-12
Reading AMBLIST Output. 15-13
Module Summary. 15-14
LISTOBJ Outputs. 15-18
LISTLOAD OUTPUT=MODLIST Output 15-28
LISTLOAD OUTPUT=XREF Output 15-37
LISTLOAD OUTPUT=XREF Output (Comparison of Load Module and Program Object Version 1) 15-44
LISTIDR Output 15-47
LISTLPA Output 15-50
LISTLOAD Output: DAT-ON Nucleus 15-51

Chapter 16. SPZAP 16-1
Planning for SPZAP 16-1
Inspecting and Modifying Data 16-2
Inspecting and Modifying a Load Module 16-2
Inspecting and Modifying a Data Record 16-10
Accessing a Data Record. 16-12
Updating the System Status Index (SSI) 16-13
Running SPZAP 16-14
Using JCL and Control Statements to Run SPZAP 16-15

Chapter 17. Dump Suppression 17-1
Using DAE to Suppress Dumps 17-1
Performing Dump Suppression 17-2
Planning for DAE Dump Suppression 17-5
Accessing the DAE Data Set 17-8
Stopping, Starting, and Changing DAE 17-10
Changing DAE Processing in a Sysplex 17-11
Using a SLIP Command to Suppress Dumps 17-11
Using an ABEND Macro to Suppress Dumps 17-12
Using Installation Exit Routines to Suppress Dumps 17-12
Determining Why a Dump Was Suppressed 17-12

Chapter 18. Messages 18-1
Producing Messages 18-1
Receiving Messages 18-2
Console. 18-2
Receiving Symptom Dumps 18-3
Planning Message Processing for Diagnosis 18-4
Controlling Message Location. 18-4

Appendix. Accessibility. A-1
Using assistive technologies A-1
Keyboard navigation of the user interface. A-1

Notices B-1
Programming Interfaces Information. B-2
Trademarks. B-3

Index X-1

Figures

2-1. Default Name Pattern for Automatically Allocated Dump Data Set 2-5
2-2. STATUS WORKSHEET Subcommand Sample Output — Dump Title 2-39
2-3. Sample Output from the STATUS SYSTEM Subcommand 2-40
2-4. Search Argument Abstract in the STATUS FAILDATA Report 2-41
2-5. System Mode Information in the STATUS FAILDATA Report 2-41
2-6. Time of Error Information in the STATUS FAILDATA Report 2-42
2-7. An Example of the SUMMARY TCBERROR Report 2-43
2-8. Sample Output from the VERBEXIT LOGDATA Subcommand 2-45
2-9. An Example of Output from the IPCS Subcommand SYSTRACE 2-47
2-10. Sample of the STATUS REGISTERS Report 2-48
2-11. Sample of the STATUS REGISTERS Report Run in z/Architecture Mode 2-49
3-1. SPFUSER Name Pattern for Automatically Allocated Dump Data Set 3-5
3-2. STATUS WORKSHEET Subcommand Sample Output — Dump Title 3-17
3-3. Sample Output from the STATUS SYSTEM Subcommand 3-17
3-4. Search Argument Abstract in the STATUS FAILDATA Report 3-18
3-5. System Mode Information in the STATUS FAILDATA Report 3-19
3-6. Time of Error Information in the STATUS FAILDATA Report 3-20
3-7. An Example of the SUMMARY TCBERROR Report 3-21
3-8. Sample Output from the VERBEXIT LOGDATA Subcommand 3-23
3-9. An Example of Output from the IPCS Subcommand SYSTRACE 3-25
3-10. Sample of the STATUS REGISTERS Report 3-26
4-1. Format of AMDSADMP Macro Instruction 4-11
4-2. Sample Console Output from the Stand-Alone Dump Program 4-18
4-3. Using AMDSADDD to Allocate and Initialize a Dump Data Set 4-26
4-4. Using AMDSADDD to Clear an Existing Dump Data Set 4-27
4-5. Using AMDSADDD to Reallocate the Dump Data Set 4-27
10-1. IBM-Supplied GTF Cataloged Procedure 10-3
10-2. GTF Storage Requirements 10-10
10-3. GTF Trace Options and Associated Trace Record Identifiers 10-35
10-4. Unformatted Control Record 10-78
10-5. Unformatted Lost Event Record 10-79
10-6. Unformatted User Trace Record Format 10-80
10-7. Header for Unformatted System Trace Record Format 10-81
11-1. Hierarchy of SYSAPPC Component Trace Options 11-27
11-2. SYSOMVS Component Trace Formatted with CTRACE COMP(SYSOMVS) 11-87
11-3. SY1 Trace Flow: Part 1 11-89
11-4. SY1 Trace Flow: Part 2 11-89
11-5. SY2 Trace Flow: Part 1 11-90
11-6. SY2 Trace Flow: Part 2 11-91
11-7. SCCOUNT Function Displaying SYSCALL Frequency 11-92
11-8. SCCOUNT Function Displaying Function Code Frequency 11-92
11-9. SYSXES SUB Trace Structure 11-132
13-1. Layout of the GFS Trace Output 13-5
14-1. Logrec Error Recording Overview 14-1
14-2. Log Stream SUBSYS Data Set Specification 14-13
15-1. Sample Module Summary for a Load Module Processed by the Linkage Editor 15-14
15-2. Sample Module Summary for a Program Object Processed by the Binder 15-15
15-3. Sample Output for LISTOBJ with an Object Module 15-18
15-4. Sample Output for LISTOBJ with XSD Record 15-19
15-5. Sample Output for LISTOBJ with GOFF Records 15-20
15-6. LISTOBJ Format for GOFF 15-22
15-7. Sample Output for LISTLOAD OUTPUT=MODLIST, ADATA=YES for a Program Object 15-28

15-8. Sample Output for LISTLOAD OUTPUT=XREF for a Program Object with Class Names: B_PRV and B_TEXT 15-38
15-9. Sample Output for LISTLOAD OUTPUT=XREF for a Load Module 15-44
15-10. Sample Output for LISTLOAD OUTPUT=XREF for a Program Object 15-45
15-11. Sample LISTIDR Output for a Load Module Processed by Linkage Editor or Binder 15-47
15-12. Sample LISTIDR Output for a Program Object Processed by Binder 15-48
15-13. Sample LISTLPA Output. 15-50
15-14. Sample Output for LISTLOAD OUTPUT=MODLIST for a PDS 15-51
15-15. Sample Output for LISTLOAD OUTPUT=MODLIST for a PDSE (Program Object Version 1) 15-54
15-16. Sample Output for LISTLOAD OUTPUT=XREF for a PDSE (Program Object Version 1) 15-55
15-17. Sample Output for LISTLOAD OUTPUT=BOTH for a 15-56
16-1. Sample Assembly Listing Showing Multiple Control Sections 16-10
16-2. SSI Bytes in a Load Module Directory Entry 16-13
16-3. Flag Bytes in the System Status Index Field 16-14
16-4. Sample Assembler Code for Dynamic Invocation of SPZAP. 16-29
16-5. Sample Formatted Hexadecimal Dump 16-31
16-6. Sample Translated Dump 16-31
16-7. Sample Formatted Hexadecimal Dump for PDSE Program Object Module 16-33
16-8. Sample Translated Dump for PDSE Data Library. 16-34

Tables

1-1. Selecting a Dump 1-1
1-2. Selecting a Trace 1-2
1-3. Selecting a Service Aid 1-2
1-4. Description of Dumps 1-3
1-5. Description of Traces 1-4
1-6. Description of Service Aids 1-5
2-1. Sample Operator DUMP Command Members in SYS1.SAMPLIB 2-14
2-2. Customizing SVC Dump Contents through the SDATA Parameter 2-22
2-3. Customizing SVC Dump Contents through Summary Dumps 2-26
2-4. Customizing SVC Dump Contents through Operator Commands 2-30
3-1. Customizing Transaction Dump Contents through the SDATA Parameter 3-8
3-2. Customizing Transaction Dump Contents through Operator Commands 3-12
4-1. DDNAMES and Defaults Used by AMDSAOSG 4-29
4-2. AMDSAOSG Return Codes 4-30
8-1. References For System Trace Entry Format Description 8-5
10-1. Combining GTF Options 10-23
10-2. GTF Trace Options and Corresponding Prompting Keywords 10-24
10-3. CCW Defaults for Selected TRACE Options 10-25
10-4. Event Identifiers and the Types of Events They Represent 10-29
11-1. FMH-5 Trace Entries in the SYSAPPC Component Trace 11-35
15-1. Program Object and Load Module Attributes 15-16

About this document

This document covers the tools and service aids that IBM provides for use in diagnosing MVS™ problems. This edition supports z/OS (5694-A01) and z/OS.e (5655-G52).

The first chapter, Chapter 1, “Selecting Tools and Service Aids” on page 1-1, contains a guide on how to select the appropriate tool or service aid for your purposes. It also provides an overview of all the tools and service aids available,

Each subsequent chapter covers one of the tools or service aids. While chapters vary, the following topics are generally covered for each tool or service aid:
- Customizing and planning information
- Starting and stopping the tool or service aid
- Receiving, formatting, and reading the output from the tool or service aid.

At the beginning of each chapter, there is a short editorial-style comment intended to characterize the tool or service aid covered in the chapter.

Who should use this document

This document is for anyone who diagnoses software problems that occur while running the operating system. This person is usually a system programmer for the installation. This document is also for application programmers who are testing their programs.

This document assumes that the reader:
- Understands basic system concepts and the use of system services
- Codes in Assembler language, and reads Assembler and linkage editor output
- Codes JCL statements for batch jobs and cataloged procedures
- Understands the commonly used diagnostic tasks and aids, such as message logs, dumps, and the interactive problem control system (IPCS)
- Understands how to search problem reporting data bases
- Understands the techniques for reporting problems to IBM

Where to find more information

Where necessary, this document references information in other documents, using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*. The following table lists titles and order numbers for documents related to other products.

For the titles and order numbers of DFP and DFSMS/MVS® documents, see the General Information manual for the version of the DFP product you have installed.

Short Title Used in This Document	Title	Order Number
	Enterprise System/9000® Models 520, 640, 660, 740, 820, 860, and 900: Recovery Guide, Volume A02	GC38-0090
Principles of Operation	*z/Architecture™ Principles of Operation*	SA22-7832
SNA Network Product Formats	*SNA Network Product Formats*	LY43-0081

Information updates on the web

For the latest information updates that have been provided in PTF cover letters and Documentation APARs for z/OS and z/OS.e, see the online document at:

`http://www.s390.ibm.com:80/bookmgr-cgi/bookmgr.cmd/BOOKS/ZIDOCMST/CCONTENTS`

This document is updated weekly and lists documentation changes before they are incorporated into z/OS publications.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for most messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

`http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/`

or from anywhere in z/OS where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS). You can also download code from the *z/OS Collection* (SK3T-4269) and the LookAt Web site that will allow you to access LookAt from a handheld computer (Palm Pilot VIIx suggested).

1. z/OS.e™ customers received a Memo to Licensees, (GI10-0684) that includes this key code.

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the **News** section on the LookAt Web site.

Some messages have information in more than one document. For those messages, LookAt displays a list of documents in which the message appears.

Summary of changes

Summary of changes
for GA22-7589-03
z/OS Version 1 Release 4

This document contains information previously presented in *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589-02, which supports z/OS Version 1 Release 3.

New information

- Information is added to indicate this document supports z/OS.e.

Changed information

- Throughout this document, improvements have been made to provide current examples, reference current hardware, and increase usability.

Moved Information

- “Planning Data Set Management for SVC Dumps” on page 2-2 has been reorganized for clarity purposes.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of changes
for GA22-7589-02
z/OS Version 1 Release 3

The book contains information previously presented in *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589-01, which supports z/OS Version 1 Release 2.

New information

- The SYSIEFAL Component Trace information for Allocation has been added.
- Minor updates to SYSRSM have been added.
- Minor updates to the SYSIOS Component Trace information have been added.
- An appendix with z/OS product accessibility information has been added.

Changed information

- Throughout the book, improvements have been made to provide current examples, reference current hardware, and increase usability.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes
for GA22-7589-01
z/OS Version 1 Release 2

The book contains information previously presented in *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589-00, which supports z/OS Version 1 Release 1.

Changed information

- Throughout the book, improvements have been made to provide current examples, reference current hardware, and increase usability.
- Information on how to capture a stand-along dump quickly has been added to Chapter 4, “Stand-Alone Dump” on page 4-1.

Deleted information

- The chapter on "Instruction Address Trace" has been removed.

This book contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes
for GA22-7589-00
z/OS Version 1 Release 1

This book contains information also presented in *OS/390® MVS Diagnosis: Tools and Service Aids*.

This book contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Chapter 1. Selecting Tools and Service Aids

This chapter introduces the tools and service aids that MVS provides for diagnosis. For the purposes of this document, **tools** includes dumps and traces, while **service aids** includes the other facilities provided for diagnosis. For example:

- SVC dump and system trace are tools.
- Logrec data set and AMBLIST are service aids.

Major Topics

There are two topics in this chapter:

- “How Do I Know Which Tool or Service Aid to Select?” - This topic lists problem types and matches them with the appropriate tool or service aid. Use this topic to select the tool or service aid you need for a particular problem.
- “What Tools and Service Aids are Available?” on page 1-3 - This topic describes each tools and service aids, including when to use it for diagnosis. Use this topic when you need an overview of tools and service aids available or to find the appropriate time to use a particular tool or service aid.

How Do I Know Which Tool or Service Aid to Select?

This topic provides criterion for selecting a tool or service aid, depending on the problem or need. There are three tables:

- Selecting a Dump, Table 1-1
- Selecting a Trace, Figure 10-7 on page 10-81
- Selecting a Service Aid, Table 1-3 on page 1-2

The tables show the problem or need, the corresponding tool or service aid, and the chapter or document that covers it in complete detail. (Most of the detailed information on tools and service aids is in this document.) Use these tables to quickly find a tool or service aid.

Table 1-1. Selecting a Dump

What is the Problem or Need?	Type of Dump to Use
Abnormal end of an authorized program or a problem program	ABEND dump See Chapter 5, “ABEND Dumps” on page 5-1 for detailed information.
Testing of a problem program while it is running	SNAP dump See Chapter 6, “SNAP Dumps” on page 6-1 for detailed information.
System problem when the system stops processing or is stopped by the operator because of slowdown or looping	Stand-alone dump See Chapter 4, “Stand-Alone Dump” on page 4-1 for detailed information.
System problem when the system continues processing	SVC dump See Chapter 2, “SVC Dump” on page 2-1 for detailed information.

Selecting Tools and Service Aids

Table 1-2. Selecting a Trace

What is the Problem or Need?	Type of Trace to Use
System problem: diagnosis requires checking of component events	Component trace See Chapter 11, “Component Trace” on page 11-1 for detailed information.
System problem: diagnosis requires detailed checking of one or two system events	Generalized trace facility (GTF) trace See Chapter 10, “The Generalized Trace Facility (GTF)” on page 10-1 for detailed information.
System or authorized program problem: diagnosis requires the messages related to a dump	Master trace See Chapter 9, “Master Trace” on page 9-1 for detailed information.
System problem: diagnosis requires checking many system events	System trace See Chapter 8, “System Trace” on page 8-1 for detailed information.
System or problem program: diagnosis requires information about allocation of virtual storage.	GETMAIN, FREEMAIN, STORAGE (GFS) trace See Chapter 13, “GETMAIN, FREEMAIN, STORAGE (GFS) Trace” on page 13-1 for detailed information.

Table 1-3. Selecting a Service Aid

What is the Problem or Need?	Type of Service Aid to Use
System or hardware problem: need a starting point for diagnosis or when diagnosis requires an overview of system and hardware events in chronological order.	Logrec data set See Chapter 14, “Recording Logrec Error Records” on page 14-1 for detailed information.
Information about the content of load modules and program objects or problem with modules on system.	AMBLIST See Chapter 15, “AMBLIST” on page 15-1 for detailed information.
Diagnosis requires dynamic change to a program, such as fixing program errors, inserting a SLIP trap match, or altering a program to start component trace.	SPZAP See Chapter 16, “SPZAP” on page 16-1 for detailed information.
Need to eliminate duplicate or unneeded dumps.	DAE See Chapter 17, “Dump Suppression” on page 17-1 for detailed information.
Diagnosis requires a trap to catch problem data while a program is running.	SLIP See *z/OS MVS System Commands* for detailed information.
Diagnosis requires formatted output of problem data, such as a dump or trace.	IPCS See *z/OS MVS IPCS User’s Guide* for detailed information.

What Tools and Service Aids are Available?

This topic provides an overview of the tools and service aids in a little more detail. The tables that follow contain a brief description of each tool or service aid, some reasons why you would use it, and a reference to the chapter or document that covers the tool or service aid in detail. (Most of the detailed information on tools and service aids is in this document.) The tools and service aids are covered in three tables:
- Description of Dumps, Table 1-4.
- Description of Traces, Table 1-5 on page 1-4.
- Description of Service Aids, Table 1-6 on page 1-5.

In each table, the dumps, traces, or service aids are listed in order by frequency of use.

Dumps

Table 1-4. Description of Dumps

Type of Dump	Description
ABEND Dump	Use an ABEND dump when ending an authorized program or a problem program because of an uncorrectable error. These dumps show: • The virtual storage for the program requesting the dump. • System data associated with the program. The system can produce three types of ABEND dumps, SYSABEND, SYSMDUMP, and SYSUDUMP. Each one dumps different areas. Select the dump that gives the areas needed for diagnosing your problem. The IBM supplied defaults for each dump are: • SYSABEND dumps - The largest of the ABEND dumps, containing a summary dump for the failing program plus many other areas useful for analyzing processing in the failing program. • SYSMDUMP dumps - Contains a summary dump for the failing program, plus some system data for the failing task. SYSMDUMP dumps are the only ABEND dumps that you can format with IPCS. • SYSUDUMP dumps - The smallest of the ABEND dumps, containing data and areas only about the failing program. **Reference:** See Chapter 5, “ABEND Dumps” on page 5-1 for detailed information.
SNAP Dump	Use a SNAP dump when testing a problem program. A SNAP dump shows one or more areas of virtual storage that a program, while running, requests the system to dump. A series of SNAP dumps can show an area at different stages in order to picture a program’s processing, dumping one or more fields repeatedly to let the programmer check intermediate steps in calculations. SNAP dumps are preformatted, you cannot use IPCS to format them. Note that a SNAP dump is written while a program runs, rather than during abnormal end. **Reference:** See Chapter 6, “SNAP Dumps” on page 6-1 for detailed information.

Table 1-4. Description of Dumps (continued)

Type of Dump	Description
Stand-Alone Dump	Use a stand-alone dump when: • The system stops processing. • The system enters a wait state with or without a wait state code. • The system enters an instruction loop. • The system is processing slowly. These dumps show central storage and some paged-out virtual storage occupied by the system or stand-alone dump program that failed. Stand-alone dumps can be analyzed using IPCS. **Reference:** See Chapter 4, "Stand-Alone Dump" on page 4-1 for detailed information.
SVC Dumps	SVC dumps can be used in two different ways: • Most commonly, a system component requests an SVC dump when an unexpected system error occurs, but the system can continue processing. • An authorized program or the operator can also request an SVC dump when they need diagnostic data to solve a problem. SVC dumps contain a summary dump, control blocks and other system code, but the exact areas dumped depend on whether the dump was requested by a macro, command, or SLIP trap. SVC dumps can be analyzed using IPCS. **Reference:** See Chapter 2, "SVC Dump" on page 2-1 for detailed information.

Traces

Table 1-5. Description of Traces

Trace	Description
Component Trace	Use a component trace when you need trace data to report an MVS component problem to the IBM Support Center. Component tracing shows processing within an MVS component. Typically, you might use component tracing while re-creating a problem. The installation, with advice from the IBM Support Center, controls which events are traced for a component. **Reference:** See Chapter 11, "Component Trace" on page 11-1 for detailed information.
GFS Trace	Use GFS trace to collect information about requests for virtual storage via the GETMAIN, FREEMAIN, and STORAGE macro. **Reference:** See Chapter 13, "GETMAIN, FREEMAIN, STORAGE (GFS) Trace" on page 13-1 for detailed information.
GTF Trace	Use a GTF trace to show system processing through events occurring in the system over time. The installation controls which events are traced. GTF tracing uses more resources and processor time than a system trace. Use GTF when you're familiar enough with the problem to pinpoint the one or two events required to diagnose your system problem. GTF can be run to an external data set as well as a buffer. **Reference:** See Chapter 10, "The Generalized Trace Facility (GTF)" on page 10-1 for detailed information.

Table 1-5. Description of Traces (continued)

Trace	Description
Master Trace	Use the master trace to show the messages to and from the master console. Master trace is useful because it provides a log of the most recently issued messages. These can be more pertinent to your problem than the messages accompanying the dump itself. **Reference:** See Chapter 9, “Master Trace” on page 9-1 for detailed information.
System Trace	Use system trace to see system processing through events occurring in the system over time. System tracing is activated at initialization and, typically, runs continuously. It records many system events, with minimal detail about each. The events traced are predetermined, except for branch tracing. This trace uses fewer resources and is faster than a GTF trace. **Reference:** See Chapter 8, “System Trace” on page 8-1 for detailed information.

Service Aids

Table 1-6. Description of Service Aids

Service Aid	Description
AMBLIST	Use AMBLIST when you need information about the content of load modules and program objects or you have a problem related to the modules on your system. AMBLIST is a program that provides lots of data about modules in the system, such as a listing of the load modules, map of the CSECTs in a load module or program object, list of modifications in a CSECT, map of modules in the LPA, and a map of the contents of the DAT-on nucleus. **Reference:** See Chapter 15, “AMBLIST” on page 15-1 for detailed information.
Common Storage Tracking	Use common storage tracking to collect data about requests to obtain or free storage in CSA, ECSA, SQA, and ESQA. This is useful to identify jobs or address spaces using an excessive amount of common storage or ending without freeing storage. Use RMF™ or the IPCS VERBEXIT VSMDATA subcommand to display common storage tracking data. **References:** • See *z/OS MVS Initialization and Tuning Guide* for detailed information on requesting common storage tracking. • See the VSM chapter of *z/OS MVS Diagnosis: Reference* for information on the IPCS VERBEXIT VSMDATA subcommand.
DAE	Use dump analysis and elimination (DAE) to eliminate duplicate or unneeded dumps. This can help save system resources and improve system performance. **Reference:** See Chapter 17, “Dump Suppression” on page 17-1 for detailed information.
IPCS	Use IPCS to format and analyze dumps, traces, and other data. IPCS produces reports that can help in diagnosing a problem. Some dumps, such as SNAP and SYSABEND and SYSUDUMP ABEND dumps, are preformatted, and are not formatted using IPCS. **Reference:** See *z/OS MVS IPCS User’s Guide* for detailed information.

Table 1-6. Description of Service Aids (continued)

Service Aid	Description
Logrec Data Set	Use the logrec data set as a starting point for problem determination. The system records hardware errors, selected software errors, and selected system conditions in the logrec data set. Logrec information gives you an idea of where to look for a problem, supplies symptom data about the failure, and shows the order in which the errors occurred. **Reference:** See Chapter 14, “Recording Logrec Error Records” on page 14-1 for detailed information.
SLIP Traps	Use serviceability level indication processing (SLIP) to set a trap to catch problem data. SLIP can intercept program event recording (PER) or error events. When an event that matches a trap occurs, SLIP performs the problem determination action that you specify: • Requesting or suppressing a dump. • Writing a trace or a logrec data set record. • Giving control to a recovery routine. • Putting the system in a wait state. **Reference:** See the SLIP command in *z/OS MVS System Commands* for detailed information.
SPZAP	Use the SPZAP service aid to dynamically update and maintain programs and data sets. For problem determination, you can use SPZAP to: • Fix program errors by replacing a few instructions in a load module or member of a partitioned data set (PDS). • Insert an incorrect instruction in a program to force an ABEND or make a SLIP trap work. • Alter instructions in a load module to start component trace. • Replace data directly on a direct access device to reconstruct a volume table of contents (VTOC) or data records that were damaged by an I/O error or program error. **Reference:** See Chapter 16, “SPZAP” on page 16-1 for detailed information.

Chapter 2. SVC Dump

SVC dump is like a burglar alarm. . . . It lets you know something's wrong and helps you pinpoint where it started.

An SVC dump provides a representation of the virtual storage for the system when an error occurs. Typically, a system component requests the dump from a recovery routine when an unexpected error occurs. However, an authorized program or the operator can also request an SVC dump when diagnostic dump data is needed to solve a problem.

An SVC dump comes in the following types, depending on how it was requested. Note that the type of dump requested determines its contents.

- **Asynchronous SVC dump (scheduled SVC dump):**

  The system issues an instruction or the caller uses a combination of parameters on the SVC dump macro invocation. SVC dump captures all of the dump data into a set of data spaces then writes the dump data from the data spaces into a dump data set. The system is available for another SVC dump upon completion of the capture phase of the dump. In an asynchronous SVC dump, the summary dump data is captured first and can be considered more useful for diagnosis.
- **Synchronous SVC dump:**

  The requester's SVC dump macro invocation issues an instruction to obtain the dump under the current task. The system returns control to the requester once the dump data has been captured into a set of data spaces. SVC dump processing then writes the dump data from the data spaces into a dump data set. The system is available for another SVC dump upon completion of the capture phase of the dump. In a synchronous SVC dump, the summary dump data is captured last.

Each SVC dump also contains a summary dump, if requested. Because dumps requested from disabled, locked, or SRB-mode routines cannot be handled by SVC dump immediately, system activity overwrites much useful diagnostic data. The summary dump supplies copies of selected data areas taken at the time of the request. Specifying a summary dump also provides a means of dumping many predefined data areas simply by specifying one option. This summary dump data is not mixed with the SVC dump because in most cases it is chronologically out of step. Instead, each data area selected in the summary dump is separately formatted and identified. IBM recommends that you request summary dump data.

Major Topics

This chapter includes information system programmers need to know about SVC dump and SVC dump processing:

- "Using Automatically Allocated Dump Data Sets" on page 2-2
- "Using Pre-Allocated Dump Data Sets" on page 2-7
- "Using Extended Sequential Data Sets" on page 2-8
- "Obtaining SVC Dumps" on page 2-11
- "Printing, Viewing, Copying, and Clearing a Pre-Allocated or SYS1.DUMPxx Data Set" on page 2-20
- "Contents of SVC Dumps" on page 2-21
- "Analyzing Summary SVC Dumps" on page 2-32
- "Analyzing an SVC Dump" on page 2-36

Reference

See *z/OS MVS Programming: Authorized Assembler Services Guide* for information any programmer needs to know about programming the SDUMP or SDUMPX macros to obtain an SVC dump:
- Deciding when to request an SVC dump
- Understanding the types of SVC dumps that MVS produces
- Designing an application program to handle a specific type of SVC dump
- Identifying the data set to contain the dump
- Defining the contents of the dump
- Suppressing duplicate SVC dumps using dump analysis and elimination (DAE)

Planning Data Set Management for SVC Dumps

SVC dump processing stores data in dump data sets that the system allocates automatically, as needed, or that you pre-allocate manually. IBM recommends the use of automatically allocated dump data sets whenever possible. Only the space required for the dump being written is allocated. The dump is written using a system-determined block size, so write time is reduced. SMS extended attributes, such as compression and striping, can be assigned to further reduce the amount of space required and the time to write.

IBM recommends using pre-allocated dump data sets only as a back up, in case the system is not able to automatically allocate a data set. Otherwise, the dump can become truncated, making error diagnosis difficult.

Using Automatically Allocated Dump Data Sets

SVC dump processing supports automatic allocation of dump data sets at the time the system writes the dump to DASD. Automatically allocated dumps will be written using the system-determined block size. The dump data sets can be allocated as SMS-managed or non-SMS-managed, depending on the VOLSER or SMS classes defined on the DUMPDS ADD command. When the system captures a dump, it allocates a data set of the correct size from the resources you specify. See “Using Extended Sequential Data Sets” on page 2-8 for DFSMS support of extended sequential data sets. Using Extended Sequential Data Sets, the maximum size of the dump can exceed the size allowed for non-SMS managed data sets.

If automatic allocation fails, pre-allocated dump data sets are used. If no pre-allocated SYS1.DUMPnn data sets are available, message IEA793A is issued, and the dump remains in virtual storage. SVC Dump periodically retries both automatic allocation and writing to a pre-allocated dump dataset until successful or until the captured dump is deleted either by operator intervention or by the expiration of the CHNGDUMP MSGTIME parameter governing message IEA793A. If you set the MSGTIME value to 0, the system will not issue the message, and it deletes the captured dump immediately.

Naming Automatically Allocated Dump Data Sets

The installation has control of the name of the data sets created by the automatic allocation function, and you can select a name-pattern to allow for dump data set organization according to your needs. The name is determined through an installation-supplied pattern on the DUMPDS command. A set of symbols is available so that you can include the following kinds of information in the names of your automatically allocated dump data sets:
- System name
- Sysplex name
- Job name

- Local and GMT time and date
- Sequence number

You can specify a name-pattern to generate any name acceptable under normal MVS data set name standards. The only requirement is that you include the sequence number symbol to guarantee each automatically allocated dump data set has a unique name.

Using Automatic Allocation of SVC Dump Data Sets

You can specify the command instructions to enable or disable automatic allocation either in the COMMNDxx parmlib member, to take effect at IPL, or from the operator console at any time after the IPL, to dynamically modify automatic allocation settings. The DUMPDS command provides the following flexibility:

- Activate automatic allocation of dump data sets
- Add or delete allocation resources
- Direct automatic allocation to SMS or non-SMS managed storage
- Deactivate automatic allocation of dump data sets
- Reactivate automatic allocation of dump data sets
- Change the dump data set naming convention

Set up automatic allocation with the following steps:

- Set up allocation authority
- Establish a name pattern for the data sets
- Define resources for storing the data sets
- Activate automatic allocation

Once automatic allocation of these SVC Dump data sets is active, allocation to a DASD volume is done starting with the first resource allocated via the DUMPDS ADD command. When allocation to that volume is no longer successful, the next resource is then used.

SVC Dump data sets can be SMS-managed or non-SMS-managed. If the DUMPDS ADD command defined SMS classes, then the allocation will first pass these classes to the ACS routines to try to allocate the SVC dump data set as SMS-managed. If this allocation is not successful for any reason, or if no SMS classes are defined, then the data set allocation will use the DASD volumes that were defined on the DUMPDS ADD command, and the SVC Dump data set will be allocated as non-SMS-managed.

SVC Dump data sets allocated as non-SMS-managed must be single volume; they can have multiple extents but they cannot span multiple volumes. Non-SMS-managed DASD does not support striping. SVC Dump data sets allocated as SMS-managed can be multi-volume only if they are allocated as striped data sets. Striping is an attribute that must be defined in the SMS classes. Striping and compression, another SMS attribute, can be used to allocate datasets that are larger than those allowed for a pre-allocated or non-SMS managed dataset.

Note: You must update automatic class selection (ACS) routines to route the intended data set into SMS-management so that it is assigned a storage class.

Setting up Allocation Authority: To allocate dump data sets automatically, the DUMPSRV address space must have authority to allocate new data sets. Do the following:

1. **Associate the DUMPSRV address space with a user ID.**

If you have RACF® Version 2 Release 1 installed, use the STARTED general resource class to associate DUMPSRV with a user ID. For this step, the RACF started procedures table, ICHRIN03, must have a generic entry.

If you have an earlier version of RACF, use the RACF started procedures table, ICHRIN03.

2. **Authorize DUMPSRV's user ID to create new dump data sets using the naming convention in the following topic.**

   With the high-level qualifier of SYS1, the data sets are considered *group* data sets. You can assign CREATE group authority to the DUMPSRV user ID within that group.

References

- See *z/OS Security Server RACF System Programmer's Guide* for information about the RACF started procedures table.
- See *z/OS Security Server RACF Security Administrator's Guide* for information on using the STARTED general resource class and on controlling creation of new data sets.

Establishing a Name Pattern: Establishing the name pattern for the dump data sets is accomplished by the DUMPDS NAME= command. Names must conform to standard data set naming conventions and are limited to 44 characters, including periods used as delimiters between qualifiers. For a complete description, see *z/OS DFSMS: Using Data Sets*. To allow meaningful names for the dump data sets, several symbols are provided that are resolved when the dump data is captured in virtual storage. For a complete list of the symbols you can use, see the explanation of DUMPDS NAME= in *z/OS MVS System Commands*.

When determining the pattern for the dump data set names, consider any automation tools you may have at your installation that work on dump data sets. Also, the automatic allocation function requires you to include the &SEQ. sequence number symbol in your data set name pattern to guarantee unique data set names. If you do not use the sequence number, the system rejects the name pattern with message IEE855I and the previous name pattern remains in effect.

By default, the system uses the name pattern **SYS1.DUMP.D&DATE..T&TIME..&SYSNAME..S&SEQ;**

The following describes the default name pattern:

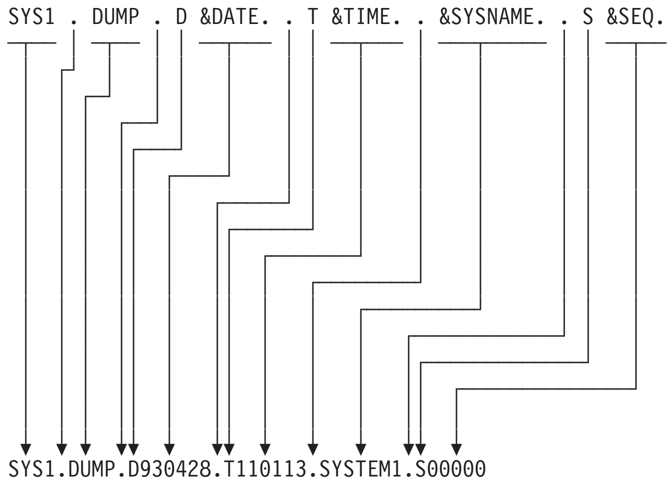

Figure 2-1. Default Name Pattern for Automatically Allocated Dump Data Set

Note: While the default data sets begin with a high-level qualifier of SYS1, this convention is no longer a requirement for data sets named by your installation.

Notice that the symbols are resolved into date, time, and sequence numerics, so they are preceded by an alphabetic character to conform to MVS data set name requirements. Also, the symbol starts with an ampersand (&) and ends with a period (.), resulting in a name pattern that has double periods when a symbol finishes a qualifier. One period ends the symbol, and the second serves as the delimiter between qualifiers of the generated data set name.

Defining Resources for Dump Data Sets: If allocation is active, SVC dump data sets can be automatically allocated as soon as resources are defined to store them. If you have not changed the name pattern, then the system default is used. See "Establishing a Name Pattern" on page 2-4. You can define dump data set resources using the DUMPDS ADD,VOL=volser (for DASD volumes) and DUMPDS ADD,SMS=class (for SMS classes) commands. You can remove resources using the DUMPDS DEL,VOL=volser and DUMPDS DEL,SMS=class commands. Automatic allocation is directed to SMS classes in preference to DASD volumes.

When automatic allocation is inactive, dumps are written to pre-allocated SYS1.DUMPxx data sets. Deactivating automatic allocation does not result in the loss of resource definitions, however. So, if automatic allocation is reactivated, all the previous resources remain available for receiving automatically allocated dump data sets. Similarly, removing the last allocation resource will not cause automatic allocation to be inactive. Removing the last allocation resource *effectively* 'turns off' the function, though, just as if all the defined resources were full. In both cases the system responds with message IEA799I and dumps are written to pre-allocated SYS1.DUMPxx data sets if they exist. Otherwise the dump remains captured until:
- You create a place for it
- The established time limit, as indicated by the CHNGDUMP MSGTIME parameter, expires
- The operator deletes the dump.

Activating Automatic Allocation: By default, automatic allocation is inactive after IPLing the system. However, you can add to your COMMNDxx parmlib member the DUMPDS NAME= command, any DUMPDS ADD commands, and the DUMPDS ALLOC=ACTIVE command to activate automatic allocation during IPL.

If you have turned off automatic allocation using ALLOC=INACTIVE, reactivate it by entering the DUMPDS ALLOC=ACTIVE operator command.

Verifying Dump Status: To verify dump status issue the DISPLAY DUMP,STATUS command. For example, after IPLing SYSTEM1 specifying DUMP=NO as a system parameter, and without requesting any dumps or specifying any DUMPDS or CHNGDUMP commands, the following output would be expected as a result of the DISPLAY DUMP,STATUS command:

```
SYSTEM1  IEE852I 10.56.03 SYS1.DUMP STATUS 204
SYS1.DUMP DATA SETS AVAILABLE=000 AND FULL=000
CAPTURED DUMPS=0000, SPACE USED=00000000M, SPACE FREE=00000500M
AUTOMATIC ALLOCATION IS: INACTIVE
   NO SMS CLASSES DEFINED
   NO DASD VOLUMES DEFINED
   NAME=SYS1.DUMP.D&DATE;.T&TIME;.&SYSNAME;.S&SEQ;
      EXAMPLE=SYS1.DUMP.D930324.T105603.SYSTEM1.S00000
```

Now assume that the following steps are performed to establish the automatic allocation function:

1. Set up your installation data set name pattern using the DUMPDS command:

   ```
   DUMPDS NAME=&SYSNAME;.&JOBNAME;.Y&YR4;M&MON;.D&DAY;T&HR;&MIN;.S&SEQ;
   ```

   Note: This step is only required if you are not using the default name pattern as shown in Figure 2-1 on page 2-5.

2. Add dump data set resources that can be used by the automatic allocation function:

   ```
   DUMPDS ADD,VOL=(SCRTH1,HSM111)
   DUMPDS ADD,SMS=(DUMPDA)
   ```

3. Activate automatic dump data set allocation using the DUMPDS command:

   ```
   DUMPDS ALLOC=ACTIVE
   ```

Note: These steps can be performed after IPL using the DUMPDS command from an operator console, or early in IPL by putting the commands in the COMMNDxx parmlib member and pointing to the member from the IEASYSxx parmlib member using CMD=xx.

If you use COMMNDxx, you may want to specify DUMP=NO in the IEASYSxx parmlib member to prevent dumps taken during IPL from being written to SYS1.DUMPxx data sets.

After issuing the DUMPDS commands shown in steps 1 through 3, requesting dump status would result in the following:

```
SYSTEM1  IEE852I 12.34.18 SYS1.DUMP STATUS 886
SYS1.DUMP DATA SETS AVAILABLE=000 AND FULL=000
CAPTURED DUMPS=0000, SPACE USED=00000000M, SPACE FREE=00000500M
AUTOMATIC ALLOCATION IS: ACTIVE
   AVAILABLE SMS CLASSES: DUMPDA
   AVAILABLE DASD VOLUMES: SCRTH1,HSM111
   NAME=&SYSNAME;.&JOBNAME;.Y&YR4;M&MON;.D&DAY;T&HR;&MIN;.S&SEQ;
      EXAMPLE=SYSTEM1.#MASTER#.Y1994M01.D26T1634.S00000
```

Managing Automatically Allocated Dump Data Sets

Automatic allocation of dump data sets is managed through the DUMPDS command. Placing appropriate commands into the COMMNDxx parmlib member allows the function to be established at IPL.

The DISPLAY DUMP command can display information about the last 100 data sets that were automatically allocated during the current IPL. Typical dump inventory management should be done using the Sysplex Dump Directory. The System Dump Directory provides access to all of the cataloged and added dump data sets created across system IPLs. Details about using the User and Sysplex Dump Directory can be found in *z/OS MVS IPCS User's Guide*.

The installation must manage the space allocated to dump data sets by limiting the volumes (non-SMS) or the classes (SMS) available for automatic allocation of dump data sets. Refer to *z/OS MVS System Commands* for the syntax of the DUMPDS ADD, DEL, and ALLOC=ACTIVE commands. For more information about SMS, see *z/OS DFSMSdss Storage Administration Reference*.

Using Pre-Allocated Dump Data Sets

To prepare your installation to receive SVC dumps, you need to provide SYS1.DUMPxx data sets. These data sets will hold the SVC dump information for later review and analysis. This section describes how to set up the SVC dump data sets, including:

- "Allocating SYS1.DUMPxx Data Sets With Secondary Extents"
- "Specifying SYS1.DUMPxx Data Sets" on page 2-10
- "Controlling SYS1.DUMPxx Data Sets" on page 2-10

Allocating SYS1.DUMPxx Data Sets With Secondary Extents

Allocate SYS1.DUMPxx data sets using the following requirements:

- Name the data set SYS1.DUMPxx, where xx is decimal 00 through 99.
- Select a device with a track size of 4160 bytes. The system writes the dump in blocked records of 4160 bytes.
- Initialize with an end of file (EOF) record as the first record.
- Allocate the data set before requesting a dump. Allocation requirements are:
  - UNIT: A permanently resident volume on a direct access device.
  - DISP: Catalog the data set (CATLG). Do not specify SHR.
  - VOLUME: Place the data set on only one volume. Allocating the dump data set on the same volume as the page data set could cause contention problems during dumping, as pages for the dumped address space are read from the page data set and written to the dump data set.
  - SPACE: An installation must consider the size of the page data set that will contain the dump data. The data set must be large enough to hold the amount of data as defined by the MAXSPACE parameter on the CHNGDUMP command, VIO pages, and pageable private area pages.

    SVC dump processing improves service by allowing secondary extents to be specified when large dump data sets are too large for the amount of DASD previously allocated. An installation can protect itself against truncated dumps by specifying secondary extents and by leaving sufficient space on volumes to allow for the expansion of the dump data sets.

    For the SPACE keyword, you can specify CONTIG to make reading and writing the data set faster. Request enough space in the primary extent to hold the smallest SVC dump expected. Request enough space in the secondary extent so that the primary plus the secondary extents can hold the largest SVC dump. The maximum size of a dataset is 65,536 tracks. For a 3390 this is 4369 cylinders, and will hold about 3 gigabytes of data. The actual size of the dump depends on the dump options in effect when the system writes the dump.

    Estimate the largest dump size as follows:

```
Bytes of SDATA options + bytes in largest region size = Result1
Result1 * number of address spaces in dump            = Result2
PLPA * 20%                                            = Result3
Bytes of requested data space storage                 = Result4

Result2 + Result3 + Result4                 = Bytes in SVC dump
```

Region size can be 16E bytes. Any applications that use above the 2-gigabyte address storage should be estimated for their use of that storage space and appropriate region size determined.

Where:
- `Result1`, `Result2`, `Result3`, `Result 4`: Intermediate results
- `SDATA options`: Described in "Contents of SVC Dumps" on page 2-21
- `PLPA`: Pageable link pack area

For the size of the smallest dump, use the default options for the SDUMPX macro. The difference between the largest dump and the smallest dump will be the size of the secondary extent.

Example: Calculating the Largest Amount of Storage

For example, to calculate the largest amount of storage required for a 3390 DASD, assume that, from the above calculations, the records needed for the SVC dump amount to 43200 kilobytes. There are 12 records per track and 15 tracks per cylinder. To determine the number of cylinders needed to allocate a data set of this size, do the following:

- For 43200 kilobytes of storage, you will need space for 10800 SVC dump records (43200 / 4 kilobytes per record).
- With 12 records per track, you will require 900 (10800 / 12 records) tracks.
- Therefore, the data set would require 60 cylinders (900 / 15 tracks per cylinder) for allocation.

Note: If you are not receiving the dump data you require, increase the size of the dump data set. You will receive system message IEA911E.

The system writes only one dump in each SYS1.DUMPxx data set. Before the data set can be used for another dump, clear it using the DUMPDS command with the CLEAR keyword.

References

- See *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU* for information about the default dump options of the SDUMPX macro.
- See *z/OS MVS System Commands* for information about using the DUMPDS command.

Using Extended Sequential Data Sets

In systems with Data Facility Storage Management Subsystem/MVS (DFSMS/MVS), IBM recommends using extended sequential data sets as dump data sets for SVC Dumps. Extended sequential data sets:
- Have a greater capacity than sequential data sets
- Support *striping*
- Support compression

Greater Capacity

Some dump data sets are quite large compared with other data sets generated by a system. An extended sequential data set can hold the largest SVC dumps, as much as 128 gigabytes.

Support for Striping

Striping spreads sections, or *stripes*, of a data set across multiple volumes and uses independent paths, if available, to those volumes. The multiple volumes and independent paths accelerate sequential reading and writing of the data set, reducing the time during which dump I/O competes with production I/O.

In a striped data set, when the last volume receives a stripe, the next stripes are placed on the first volume, the second volume, the third, and so on to the last volume, then back to the first volume. If *n* volumes are used, striping allows sequential access to the data set at nearly *n* times the rate at which a single volume data set can be processed. The faster processing speeds up moving dump data from relatively expensive data space storage to less expensive DASD.

Support for Compression

Compression allows dump data sets to use less DASD space. Before using compression, consider the following:

- Compression and decompression trade off processing cycles for more efficient use of DASD. If hardware compression is not available, the number of processing cycles is significantly higher.
- A compressed extended sequential data set cannot be updated. Therefore, IPCS cannot add secondary symptoms generated during dump analysis into the header record of a compressed dump.

  This restriction has the most impact for an SVC dump requested by an operator DUMP command or for a stand-alone dump.

Finding Automatically Allocated Dump Data Sets

The AUTODSN= parameter of the DISPLAY DUMP,TITLE operator command enables you to list up to 100 of the most recent dump data sets that were automatically allocated during this IPL. No information is preserved about data sets that were automatically allocated before the last 100. As an example, if you wanted to see the titles of the last 5 automatically allocated dump data sets, you would issue:

```
DISPLAY DUMP,TITLE,AUTODSN=5
```

For complete information on the use of the DISPLAY DUMP command, see DISPLAY DUMP in *z/OS MVS System Commands*.

Communication from the System

The system communicates about automatic allocation of dump data sets using two messages:

- IEA611I is issued when a complete or partial dump is taken to an automatically allocated dump dataset. IEA611I is an informational message, it will not be issued highlighted.
- IEA799I is issued once per captured dump when automatic allocation fails; it will not be re-issued as a result of automatic allocation failing for subsequent attempts to allocate the same dump data set unless the reason text is different.

Specifying SYS1.DUMPxx Data Sets

When planning SYS1.DUMPxx data sets, remember that the data sets frequently contain sensitive data (user or installation confidential information, logon passwords, encryption keys, etc.). Protect these data sets with RACF to limit access to them.

The installation can specify SYS1.DUMPxx data sets in two ways:

- IBM recommends that you use the DUMPDS operator command through the COMMNDxx parmlib member. Use the DUMPDS ADD command within the COMMNDxx parmlib member to ensure that all interaction with the dump data set occurs through the DUMPDS command.

Example: Adding a SYS1.DUMP Data Set

To specify data set SYS1.DUMP05, enter:

```
COM='DUMPDS ADD,DSN=05'
```

- During system initialization, in the DUMP parameter in the IEASYSxx parmlib member.

  Specify DUMP=NO in the IEASYSxx parmlib member. Otherwise, all available data sets will be allocated before the COMMNDxx parmlib member is processed.

The data sets are on direct access only. The maximum number of SYS1.DUMPxx data sets an installation can have is 100. The direct access data set must be on a permanently resident volume; that is, the data set must be allocated and cataloged. These dump data sets cannot be shared by more than one system.

All dump data sets should not be on the same pack. A pack should contain enough storage to allow the dump data sets to allocate secondary extent space, if needed.

References

- See *z/OS MVS Initialization and Tuning Reference* for information about the IEACMDxx and IEASYSxx parmlib member.
- See *z/OS MVS System Commands* for information about using the DUMPDS command.

Controlling SYS1.DUMPxx Data Sets

After system initialization, use the following to change and control these data sets:

- Copy the dump from the SYS1.DUMPxx data set to another data set; then clear the SYS1.DUMPxx data set, so that it can be sued for another dump. You can use IPCS to format and view or print the copied dump, as described in the following topic.
- Use the DUMPDS operator command to:
  - Add more SYS1.DUMPxx data sets on direct access for SVC dumps.
  - Delete SYS1.DUMPxx data sets for SVC dumps.
  - Clear a SYS1.DUMPxx data set containing a dump by writing an EOF mark as the first record. An EOF mark as the first record makes the data set available for another dump.

  A reIPL is not necessary when adding, deleting, or clearing a data set with the DUMPDS operator command.
- Use the REPLY command to system message IEA793A to cancel a dump.

- Use a post dump exit routine to copy the dump to another data set. IEAVTSEL is an SVC dump post dump exit name list that lists the module names of installation exit routines to be given control when dump processing ends.

References

- See *z/OS MVS Installation Exits* for more information about the IEAVTSEL post dump exit name list.
- See *z/OS MVS IPCS Commands* for the IPCS COPYDUMP subcommand.
- See *z/OS MVS System Commands* for the DUMPDS and REPLY operator commands.

Obtaining SVC Dumps

Obtain an SVC dump by issuing a SDUMP or SDUMPX macro in an authorized program, entering a DUMP or SLIP command, or using a SLIP command in the IEASLPxx parmlib member.

An SVC dump is written to an SVC dump data set, which is specified on the DCB parameter of the SDUMP or SDUMPX macro, available as SYS1.DUMPxx, or automatically allocated. If a SYS1.DUMPxx data set is not available, the dump is captured and the system asks the operator to provide a SYS1.DUMPxx data set. An operator can also determine the current options under which SVC dump is processing and the status of SVC dump.

Note: The limit to the number SVC dumps that can be captured is subject to the MAXSPACE value.

In a sysplex, you probably need dumps from more than one system to collect all of the problem data. These dumps need to be requested at the same time. To request these multiple dumps, do one of the following on any of the systems that might be involved in the problem:

- Enter a DUMP command with a REMOTE parameter.
- Issue a SDUMPX macro with a REMOTE parameter.
- Create a SLIP trap in an IEASLPxx parmlib member in the shared SYS1.PARMLIB or in the parmlib on each system. Because you may not be able to predict which system will first have the problem, use a ROUTE operator command to activate the traps on all systems that are similar. Each trap should include a REMOTE parameter to dump all the other systems that might be involved.

To help you set up these requests, the commands and macro can contain wildcards. If the installation gives names that form patterns to the systems in the sysplex and to jobs for associated work, you can use wildcards, * and ?, to specify the names. For example, use the name TRANS? for the jobnames TRANS1, TRANS2, and TRANS3 and the name TRANS* for TRANS1, TRANS12, and TRANS123.

This section describes each of these topics, as follows:

- “Issuing a Macro for SVC Dump” on page 2-12
- “Operator Activities” on page 2-12
- “Making a Dump Data Set Available” on page 2-15
- “Determining Current SVC Dump Options and Status” on page 2-16
- “Finding SVC Dumps” on page 2-17

Issuing a Macro for SVC Dump

In an authorized program, use an SDUMP or SDUMPX macro to request an SVC dump. The system writes the dump in a SYS1.DUMPxx data set or, if specified in the macro, in a user-supplied data set.

Example: Dumping Default Contents

To dump the default contents listed in “Contents of SVC Dumps” on page 2-21 to a SYS1.DUMPxx data set:

```
SDUMPX
```

If the dump is written to a user-supplied SVC dump data set, the program provides a data control block (DCB) for the data set, opens the DCB before issuing the SDUMP or SDUMPX macro, and closes the DCB after the dump is written. For a synchronous dump, the close should occur when the system returns control to the requester. For a scheduled dump, the close should occur when the ECB is posted or when the SRB is scheduled.

Example: Requesting a Synchronous Dump

To write a synchronous dump to a data set whose DCB address is in register 3:

```
SDUMPX DCB=(3)
```

Reference

See *z/OS MVS Programming: Authorized Assembler Services Guide* for information about requesting a scheduled SVC dump and a synchronous SVC dump.

Operator Activities

From a console with master authority, the operator can enter either of the following commands:

- DUMP operator command.

Example: Using the DUMP Command

The following operator command will write an SVC dump:
– To a SYS1.DUMPxx data set
– With a dump title of “MYDUMP1 5-9-88”
– With the default contents listed in “Contents of SVC Dumps” on page 2-21
– For a job named MYJOB1

```
DUMP COMM=(MYDUMP1 5-9-88)
```

The system will respond with the message:

```
* 23 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
```

Ask the operator to reply:

```
REPLY 23,JOBNAME=MYJOB1
```

Note that if the operator replies `REPLY 23,U` to IEE094D, the system dumps the current address space, which is the master scheduler address space. The operator must use an ASID, JOBNAME, or TSONAME parameter in the reply to obtain other dumps.

Use the DUMPDS command to produce a scheduled SVC dump.

- SLIP operator command with an ACTION option of STDUMP, SVCD, SYNCSVCD, or TRDUMP.

Example: Using the SLIP Command

The following operator command will write an SVC dump:
– To a SYS1.DUMPxx data set
– When a program check interruption occurs in a job named MYJOB1
– With the default contents shown in “Contents of SVC Dumps” on page 2-21

```
SLIP SET,ACTION=SVCD,ERRTYP=PROG,JOBNAME=MYJOB1,END
```

The SLIP command produces a scheduled SVC dump.

Operator Command in an IEASLPxx Parmlib Member

The installation can also place SLIP operator commands in IEASLPxx parmlib members to produce an SVC dump. When a command is needed, the operator dynamically sets the IEASLPxx member containing the needed SLIP command. The installation can place SLIP commands that request different types of SLIP traps in different IEASLPxx members.

References

- See *z/OS MVS System Commands* for the DUMP and SLIP operator commands.
- See *z/OS MVS Initialization and Tuning Reference* for the IEASLPxx member.

Operator Command in an IEADMCxx Parmlib Member

IEADMCxx enables you to supply DUMP command parameters through a parmlib member. IEADMCxx enables the operator to specify the collection of dump data by issuing a DUMP command, indicating the name of the parmlib member and any symbolic substitution variables.

In z/OS Release 2, a DUMP command parmlib library provides a large set of sample parmlib members that can be used when gathering dump information for IBM Service. Each of the parmlib members can be used as-is, or can be used as a base for further modification to deal with installation specific requirements, such as system names, address space names, and so on. In order to take advantage of these parmlib members, it is recommended that they be copied to a data set in your parmlib concatenation.

The DUMP command parmlib members are delivered in SYS1.SAMPLIB. This allows the parmlib members to be copied, and modified if necessary, to be used as directed by IBM Service. Care has been made to ensure that substitution is used where names can vary by installation.

Note: The substitution variable length may not be sufficient for the values used at a particular site. For example, &job in IEADMCAS can only accommodate job names that are 4 characters or less. For this example, after copying it to a parmlib member, change &job to a longer variable which can accommodate up to eight characters, like &thisjob.

The following table summarizes the sample dump commands that z/OS R2 supplies in SYS1.SAMPLIB.

Table 2-1. Sample Operator DUMP Command Members in SYS1.SAMPLIB

Member Name	Suspected Problem Area	Areas Dumped	Symbolics Used	Remote Option Used
IEADMCAR	APPC	APPC transaction environment, including RRS		
IEADMCAS	Shared Tape	Allocation Autoswitch and XCF, with affected job	&job	Y
IEADMCCA	Catalog	Catalog address space and associated areas		
IEADMCCN	Console	CONSOLE address space and its IEEMCSn data spaces		
IEADMCCP	CP/SM	CICSplex SM environment on all systems in the sysplex. This includes the CAS, CMAS and EYU address spaces.		Y
IEADMCD2	DB2® distributed transactions	DB2/RRS environment		Y
IEADMCJ2	JES2	JES2/XCF environment on current and specified system	&system	Y
IEADMCLC	CICS®	System Logger, RLS and CICS		
IEADMCLG	Logger	System Logger and GRS, on all systems in the sysplex		Y
IEADMCLS	Logger	Logger and XCF, along with specified structure name	&str	Y

Table 2-1. Sample Operator DUMP Command Members in SYS1.SAMPLIB (continued)

Member Name	Suspected Problem Area	Areas Dumped	Symbolics Used	Remote Option Used
IEADMCLX	Logger	System Logger and XCF, on all systems in the sysplex		Y
IEADMCRL	RRS	RRS and the System Logger on all systems in the sysplex		Y
IEADMCRR	RRS	RRS and its data spaces		
IEADMCSQ	IMS™	IMS Shared Queues environment (IMS Control region, CL/I SAS Region, DBRC Region, and all of the CQS address spaces connected to the shared queues		Y
IEADMCTA	TCP/IP	TCP/IP, along with the specified application	&tcp &appl	
IEADMCTC	TCP/IP	TCP/IP, along with the Comm Server address space	&tcp	
IEADMCTI	TCP/IP	TCP/IP and its data space	&tcp	
IEADMCTO	TCP/IP	TCP/IP and OMVS	&tcp	
IEADMCVC	Comm Server	VTAM® and TCP/IP, with the TCPIP and VTAM data spaces	&tcp &net	
IEADMCVG	VTAM GR	VTAM Generic Resources environment, with its CF structure		Y
IEADMCVT	Comm Server	VTAM and TCP/IP (address spaces only)	&tcp &net	
IEADMCVV	VTAM	VTAM and the VIT data space	&net	
IEADMCWL	WLM	WLM on all systems in the sysplex		Y
IEADMCWS	Web server	HTTP web server with OMVS		
IEADMCWT	Web server	HTTP web server and TCP/IP		
IEADMCXI	IRLM	XCF and IRLM		
IEADMCX1	IRLM	XCF and IRLM on all systems in the sysplex		Y

Notes:

1. When specifying parmlib members containing symbolic parameters, you must specify the symbolic and substitution value using the SYMDEF keyword.
2. The dump command indicated by each row with a “Y” in the “Remote Option Used” column results in a multi-system dump.

Making a Dump Data Set Available

An SVC dump is taken to an SVC dump data set, either specified on the DCB parameter of the SDUMP or SDUMPX macro, available as SYS1.DUMPxx, or automatically allocated. SVC dump processing issues message IEA793A when the dump has been captured but there are no available dump data sets. When a SYS1.DUMPxx data set is not available, the operator has the option either of deleting the captured dump by replying D or making another dump data set available to SVC dump processing. To make another dump data set available, the operator uses the DUMPDS command.

Example: Using the DUMPDS Command

Use a DUMPDS command to make a dump data set available to SVC dump.

System message:

```
* 16 IEA793A NO DUMP SVC DUMP DATA SETS AVAILABLE FOR DUMPID=123.
* 16 IEA793A USE THE DUMPDS COMMAND OR REPLY D TO DELETE THE
             CAPTURED DUMP
```

Operator reply:

```
DUMPDS ADD,DSN=02
```

References

- See *z/OS MVS System Commands* for information about the DUMPDS command.
- See *z/OS MVS System Messages, Vol 6 (GOS-IEA)* for information about message IEA793A.

Determining Current SVC Dump Options and Status

An operator can determine the current dump options and the SYS1.DUMPxx data sets that contain SVC dumps.

Dump Mode and Options

Use a DISPLAY DUMP operator command to get the dump mode and options in effect for SVC dumps and SYSABEND, SYSMDUMP, and SYSUDUMP dumps. The system displays the mode and options in message IEE857I.

References

- See *z/OS MVS System Commands* for the DISPLAY for more information about dump modes.
- See *z/OS MVS System Messages, Vol 7 (IEB-IEE)* for more information about IEE857I.

Example: Determining the Mode and Options

To ask for the mode and options, enter:

```
DISPLAY DUMP,OPTIONS
```

If the options listed are not the ones desired, have the operator use a CHNGDUMP operator command to change them.

Status of SYS1.DUMPxx Data Sets

Use a DISPLAY DUMP operator command to get the status of all defined SYS1.DUMPxx data sets on direct access. The system displays the status in message IEE852I or IEE856I. The message indicates the full and available data sets.

Example: Determining the Status

To ask for the status of SYS1.DUMPxx data sets, enter:

```
DISPLAY DUMP,STATUS
```

References

- See *z/OS MVS System Commands* for the CHNGDUMP and DISPLAY commands.
- For a description of these messages, use LookAt or see *MVS System Messages*. For a description of LookAt, see “Using LookAt to look up message explanations” on page xviii.

Finding SVC Dumps

An operator can search the current SYS1.DUMPxx data sets for the SVC dump for a particular problem. To select the dump, use the title and time or use the dump symptoms. The operator can also find a dump that has been captured in virtual storage but has not been written to a data set.

Title and Time of SVC Dump(s)

Use one of the following to get the titles and times for SVC dumps:

- A DISPLAY DUMP operator command entered through a console with master authority. The system displays the titles and times in message IEE853I.

Example: Finding Title and Time Using DISPLAY

To see the titles and times for the dumps in SYS1.DUMP08 and SYS1.DUMP23, without displaying any automatically allocated dump data sets, enter:

```
DISPLAY DUMP,TITLE,DSN=(08,23)
```

To display the titles of the most recently automatically allocated dump data set and all pre-allocated dump data sets, enter:

```
DISPLAY DUMP,TITLE
```

or:

```
DISPLAY DUMP,TITLE,DSN=ALL
```

To display the titles of the last 5 most recently allocated dump data sets, enter:

```
DISPLAY DUMP,TITLE,AUTODSN=5
```

To see the dump titles for all captured dumps, enter:

```
DISPLAY DUMP,TITLE,DUMPID=ALL
```

- An IPCS SYSDSCAN command entered at a terminal by a TSO/E user. IPCS displays the titles and times at the terminal.

Example: Finding the Title and Time Using IPCS

To see the dump titles and times for the dumps in SYS1.DUMP08 and SYS1.DUMP23, enter the following IPCS command:

```
SYSDSCAN 08
SYSDSCAN 23
```

Reference

See *z/OS MVS IPCS Commands* for information about SYSDSCAN.

If a data set listed in either command is empty or undefined, the system issues a message to tell why the title is not available.

Symptoms from SVC Dumps

Use a DISPLAY DUMP operator command to get the symptoms from SVC dumps in SYS1.DUMPxx data sets on direct access or from SVC dumps that have been captured in virtual storage. The system displays the following symptoms in message IEE854I:

- Dump title or a message telling why the title is not available
- Error id consisting of a sequence number, the processor id, the ASID for the failing task, and the time stamp
- System abend code
- User abend code
- Reason code
- Module name
- Failing CSECT name
- Program status word (PSW) at the time of the error
- Interrupt length code in the system diagnostic work area (SDWA)
- Interrupt code in the SDWA
- Translation exception address in the SDWA
- Address of the failing program in the SDWA
- Address of the recovery routine in the SDWA
- Registers at the time of the error saved in the SDWA

Example: Viewing Symptoms

To see symptoms from the dump in the SYS1.DUMP03 data set without displaying any automatically allocated data sets, enter:

```
DISPLAY DUMP,ERRDATA,DSN=03
```

To see symptoms from the most recently automatically allocated dump data set and all pre-allocated dump data sets, enter:

```
DISPLAY DUMP,ERRDATA
```

or:

```
DISPLAY DUMP,ERRDATA,DSN=ALL
```

To see symptoms from the last 5 most recently allocated dump data sets, enter:

```
DISPLAY DUMP,ERRDATA,AUTODSN=5
```

To see symptoms from the captured dump identified by DUMPID=005, enter:

```
DISPLAY DUMP,ERRDATA,DUMPID=005
```

Example: Output from DISPLAY,DUMP ERRDATA Command

Using the DISPLAY DUMP,ERRDATA command, you can retrieve basic information about the dump without having to format the dump or read through the system log. Message IEE854I displays the error data information, including the PSW at the time of the error, the system abend code and reason code, and the module and CSECT involved.

```
S430  d d,errdata
S430  IEE854I 13.01.25 SYS1.DUMP ERRDATA 745
SYS1.DUMP DATA SETS AVAILABLE=001 AND FULL=001
CAPTURED DUMPS=0000, SPACE USED=00000000M, SPACE FREE=00000500M
     DUMP00 TITLE=ABDUMP ERROR,COMPON=ABDUMP,COMPID=5752-SCDMP,
                  ISSUER=IEAVTABD
         DUMP TAKEN TIME=13.01.02 DATE=09/27/1996
         ERRORID=SEQ00010 CPU0000 ASID0010 TIME=13.00.55
         SYSTEM ABEND CODE=0C1  REASON CODE=00000001
         MODULE=IGC0101C CSECT=IEAVTABD
         PSW AT TIME OF ERROR=070C0000 823FE0F6 ILC=2 INT=01
         TRANSLATION EXCEPTION ADDR=008E1094
         ABENDING PROGRAM ADDR=******** RECOVERY ROUTINE=ADRECOV
         GPR 0-3   00000000  7F6EBAE4  023FFFF6  023F2BFF
         GPR 4-7   008FD088  023FEFF7  823FDFF8  7F6EC090
         GPR 8-11  00000048  7F6EC938  7F6EBA68  7F6EBA68
         GPR12-15  7F6EB538  7F6EB538  00000000  00000000
NO DUMP DATA AVAILABLE FOR THE FOLLOWING EMPTY SYS1.DUMP DATA SETS: 01
```

Reference

See *z/OS MVS System Commands* for the DISPLAY operator command.

Printing, Viewing, Copying, and Clearing a Pre-Allocated or SYS1.DUMPxx Data Set

SVC dumps are unformatted when created. Use IPCS to format a dump and then view it at a terminal or print it.

After the dump has been copied to a permanent data set, use a DUMPDS operator command to clear the data set so that the system can use the data set for another dump. Then use IPCS to view the copy.

You can copy a dump that was written to tape so that you can view the dump through IPCS more efficiently.

Example: JCL to Print, Copy, and Clear an SVC Dump Data Set

For a pre-allocated data set or a SYS1.DUMPxx data set, this JCL does the following:

- Uses the SVC dump in the SYS1.DUMP00 data set. The IPCSDUMP DD statement identifies this data set.
- Copies the dump from the SYS1.DUMP00 data set to the data set identified in the DUMPOUT DD statement. To use this example, change the DUMPOUT DD statement to give the DSNAME for the desired location.
- Clears the SYS1.DUMP00 data set so that it can be used for a new dump.
- Deletes the IPCS dump directory in the DELETE(DDIR) statement. This statement uses the USERID of the batch job in the directory identification.
- Allocates the dump directory through the BLSCDDIR statement. The default is volume VSAM01. The example shows VSAM11. Override the default volume with the desired volume.
- Formats the dump using the IPCS subcommands in LIST 0. To use this example, replace the LIST 0 command with the desired IPCS subcommands or a CLIST. See *z/OS MVS IPCS User's Guide* for CLISTs.

```
//IPCSJOB  JOB
//IPCS     EXEC PGM=IKJEFT01,DYNAMNBR=75,REGION=1500K
//SYSPROC  DD   DSN=SYS1.SBLSCLI0,DISP=SHR
//IPCSDUMP DD   DSN=SYS1.DUMP00,DISP=SHR
//DUMPOUT  DD   DSN=GDG.DATA.SET(+1),DISP=SHR
//SYSUDUMP DD   SYSOUT=*
//IPCSTOC  DD   SYSOUT=*
//IPCSPRNT DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
DELETE(DDIR) PURGE CLUSTER
BLSCDDIR VOLUME(VSAM11)
IPCS NOPARM
SETDEF DD(IPCSDUMP) LIST NOCONFIRM
LIST 0
COPYDUMP INFILE(IPCSDUMP) OUTFILE(DUMPOUT) CLEAR NOPRINT NOCONFIRM
END
/*
```

Contents of SVC Dumps

Unlike ABEND dumps, SVC dumps do not have a parmlib member that establishes the dump options list at system initialization. The IBM-supplied IEACMD00 parmlib member contains a CHNGDUMP operator command that adds the local system queue area (LSQA) and trace data (TRT) to every SVC dump requested by an SDUMP or SDUMPX macro or a DUMP operator command, but not for SVC dumps requested by SLIP operator commands.

The contents of areas in an SVC dump depend on the dump type:

- Scheduled SVC dump: The current task control block (TCB) and request block (RB) in the dump are for the dump task, rather than for the failing task. For additional address spaces in the dump, the TCB and RB are for the dump task.
- Synchronous SVC dump: The current TCB and RB in the dump are for the failing task.

Reference

See *z/OS MVS IPCS Commands* for examples of IPCS output formatted from SVC dumps.

Customizing SVC Dump Contents

You can customize the contents of an SVC dump to meet the needs of your installation. For example, you might want to add areas to be dumped, reduce the dump size, or dump Hiperspaces. In most cases, you will customize the contents of an SVC dump or summary dump through the SDATA parameter of the SDUMP or SDUMPX macro or through operator commands.

Reducing Dump Size

To obtain a smaller dump that does not have all the usual defaults, code a NODEFAULTS option in the SDATA parameter of the SDUMP or SDUMPX macro. With the NODEFAULTS option, the dump contains:

- Certain default system areas needed by IPCS for dump analysis
- Areas requested on the SDUMP or SDUMPX macro

Hiperspaces

SVC dumps do not include Hiperspaces. To include Hiperspace™ data in an SVC dump, you have to write a program to copy data from the Hiperspace into address space storage that is being dumped.

Adding Areas

If the dump, as requested, will not contain all the needed areas, see one of the following for ways to add the areas:

- “Customized Contents Using the SDATA Parameter”
- “Contents of Summary Dumps in SVC Dumps” on page 2-26
- “Customizing Contents Through Operator Commands” on page 2-29

Customized Contents Using the SDATA Parameter

The IBM-supplied default contents and the contents available through customization are detailed in Table 2-2 on page 2-22. The tables show dump contents alphabetically by the parameters that specify the areas in the dumps. Before requesting a dump, decide what areas will be used to diagnose potential errors. Find the areas in the tables. The symbols in columns under the dump indicate how the area can be obtained in that dump. The symbols are:

C Available on the command that requests the dump
D IBM-supplied default contents
M Available on the macro that requests the dump
P Available in the parmlib member that controls the dump options
X Available on the CHNGDUMP operator command that changes the options for the dump type
blank No symbol indicates that the area cannot be obtained.

Note: System operator commands and assembler macros use the parameters in the table to specify dump contents.

The order of the symbols in the following table is not important.

Table 2-2. Customizing SVC Dump Contents through the SDATA Parameter

SDATA Parameter Option	Dump Contents	SVC Dump for SDUMP or SDUMPX Macro or DUMP Command with SDATA Parameter	SVC Dump for DUMP Command without SDATA Parameter	SVC Dump for SLIP Command ACTION= SVCD or SYNCSVCD	SVC Dump for SLIP Command ACTION= STDUMP	SVC Dump for SLIP Command ACTION= TRDUMP	SVC Dump for DUMP Command SVCDUMPRGN= YES
ALLNUC	The DAT-on and DAT-off nucleuses	M C X	X	C	C	C	
ALLPSA	Prefixed save area (PSA) for all processors	D M X	D X	D C	C	C	
COUPLE	Data on cross-system coupling **Note:** COUPLE cannot be specified on an SDUMP macro. It can, however, be specified on an SDUMPX macro.	M C X		C	C	C	
CSA	Common service area (CSA) (that is, subpools 227, 228, 231, 241)	M C X	D	D C	C	C	
DEFAULTS	Default areas	M X					
GRSQ	Global resource serialization control blocks for the task being dumped: • Global queue control blocks • Local queue control blocks	M C X	X	C	C	C	

Table 2-2. Customizing SVC Dump Contents through the SDATA Parameter (continued)

SDATA Parameter Option	Dump Contents	SVC Dump for SDUMP or SDUMPX Macro or DUMP Command with SDATA Parameter	SVC Dump for DUMP Command without SDATA Parameter	SVC Dump for SLIP Command ACTION= SVCD or SYNCSVCD	SVC Dump for SLIP Command ACTION= STDUMP	SVC Dump for SLIP Command ACTION= TRDUMP	SVC Dump for DUMP Command SVCDUMPRGN= YES
IO	Input/output supervisor (IOS) control blocks for the task being dumped: • EXCPD • UCB	D					
LPA	Active link pack area (LPA): module names and contents	M C X	X	D C	C	C	
LSQA	Local system queue area (LSQA) allocated for the address space (that is, subpools 203 - 205, 213 - 215, 223 - 225, 229, 230, 233 - 235, 249, 253 - 255)	D M C X	D X	D C	C	C	
NOALL	No ALLPSA	M X	X	C	C	C	
NOALLPSA	No ALLPSA	M X	X	C	D C	D C	
NODEFAULTS	• Minimum default areas needed for IPCS dump analysis • Areas requested on the SDUMP or SDUMPX macro	M					
NOPSA	No PSA	C					
NOSQA	No SQA	M C X	X	C	D C	D C	
NOSUM	No SUM	M C X	X	C	D C	D C	
NUC	Read/write portion of the control program nucleus (that is, only the non-page -protected areas of the DAT-on nucleus), including: • CVT • LSQA • PSA • SQA	M C X	X	D C	C	C	

Table 2-2. Customizing SVC Dump Contents through the SDATA Parameter (continued)

SDATA Parameter Option	Dump Contents	SVC Dump for SDUMP or SDUMPX Macro or DUMP Command with SDATA Parameter	SVC Dump for DUMP Command without SDATA Parameter	SVC Dump for SLIP Command ACTION= SVCD or SYNCSVCD	SVC Dump for SLIP Command ACTION= STDUMP	SVC Dump for SLIP Command ACTION= TRDUMP	SVC Dump for DUMP Command SVCDUMPRGN= YES
PSA	Prefixed save areas (PSA) for the processor at the time of the error or the processor at the time of the dump	D M C X	D X	D C	C	C	
RGN	Allocated pages in the private area of each address space being dumped, including subpools 0 - 127, 129 - 132, 203 - 205, 213 - 215, 223 - 225, 229, 230, 236, 237, 244, 249, 251 - 255. Also, allocated eligible storage above the 2–gigabyte address.	M C X	X	D C	C	C	C
SERVERS	Areas added by IEASDUMP. SERVERS exits	M C X					
SQA	System queue area (SQA) allocated (that is, subpools 226, 239, 245, 247, 248)	D M C X	D X	D C	C	C	
SUM	Summary dump (See "Contents of Summary Dumps in SVC Dumps" on page 2-26.)	D M C X	D X	D C	C	C	
SWA	Scheduler work area (SWA) (that is, subpools 236 and 237)	M C X	D X	C	C	C	
TRT	System trace, generalized trace facility (GTF) trace, and master trace, as available	D M C X	D X	D C	D C	D C	
Default system data	Instruction address trace, if available	D	D	D	D	D	

Table 2-2. Customizing SVC Dump Contents through the SDATA Parameter (continued)

SDATA Parameter Option	Dump Contents	SVC Dump for SDUMP or SDUMPX Macro or DUMP Command with SDATA Parameter	SVC Dump for DUMP Command without SDATA Parameter	SVC Dump for SLIP Command ACTION= SVCD or SYNCSVCD	SVC Dump for SLIP Command ACTION= STDUMP	SVC Dump for SLIP Command ACTION= TRDUMP	SVC Dump for DUMP Command SVCDUMPRGN= YES
Default system data	Nucleus map and system control blocks, including: • ASCB for each address space being dumped • ASVT • Authoriza- tion table for each address space • CVT, CVT prefix, and secondary CVT (SCVT) • Entry tables for each address space • GDA • JSAB of each address space being dumped • Linkage stack • Linkage table for each address space • PCCA and the PCCA vector table • TOT • TRVT • UCB	D					
Default system data	DFP problem data, if DFP Release 3.1.0 or a later release is installed	D	D	D	D	D	
Default system data	Storage for the task being dumped and program data for all of its subtasks	D	D	D			
Default system data	Storage: 4 kilobytes before and 4 kilobytes after the address in the PSW at the time of the error	D					
Default system data	SUBTASKS: Storage for the task being dumped and program data for all of its subtasks		D	D			

Contents of Summary Dumps in SVC Dumps

Request a summary dump for two reasons:

1. The SUM or SUMDUMP parameters request many useful, predefined areas with one parameter.
2. The system does not write dumps immediately for requests from disabled, locked, or SRB-mode programs. Therefore, system activity destroys much needed diagnostic data. When SUM or SUMDUMP is specified, the system saves copies of selected data areas at the time of the request, then includes the areas in the SVC dump when it is written.

Use SDUMP or SDUMPX macro parameters to request different types of summary dumps, as follows:

- **Disabled Summary Dump**: This summary dump saves data that is subject to rapid and frequent change before returning control to the scheduled dump requester. Because the system is disabled for this dump, the dump includes only data that is paged in or in DREF storage. Specify BRANCH=YES and SUSPEND=NO on an SDUMP or SDUMPX macro to obtain a disabled summary dump.
- **Suspend Summary Dump**: This summary dump also saves data that is subject to rapid and frequent change before returning control to the scheduled dump requester. This dump, however, can save pageable data. To obtain a suspend summary dump, do the following:
  - For an SDUMP or SDUMPX macro, specify BRANCH=YES and SUSPEND=YES
  - For an SDUMPX macro, specify BRANCH=NO for a scheduled dump with SUMLSTL parameter
- **Enabled Summary Dump**: This summary dump does not contain volatile system information. The system writes this summary dump before returning control to the dump requester; the summary information is saved for each address space being dumped. To obtain an enabled summary dump, do the following:
  - For an SDUMP or SDUMPX macro, specify BRANCH=NO.
  - For a SLIP operator command, do not specify an SDATA parameter or specify SUM in an SDATA parameter.
  - For a DUMP operator command, do not specify an SDATA parameter or specify SUM in an SDATA parameter. Note that this dump does not contain data that the system creates when it detects a problem; for example, this dump would not contain a system diagnostic work area (SDWA).

In the following table, an S indicates that the area is included in the summary dump for the dump type.

Table 2-3. Customizing SVC Dump Contents through Summary Dumps

Summary Dump Contents	Disabled Summary Dump	Suspend Summary Dump	Enabled Summary Dump
Address space identifier (ASID) record for the address space of the dump task			S

Table 2-3. Customizing SVC Dump Contents through Summary Dumps (continued)

Summary Dump Contents	Disabled Summary Dump	Suspend Summary Dump	Enabled Summary Dump
Control blocks for the failing task, including: • For task-mode dump requesters: – Address space control block (ASCB) – Request blocks (RB) – System diagnostic work areas (SDWA) pointed to by the recovery termination management 2 work areas (RTM2WA) associated with the task control block (TCB) – TCB – Extended status block (XSB) • For service request block (SRB)-mode dump requesters: – ASCB – Suspended SRB save area (SSRB) – SDWA used for dump – XSB	S	 S	
Control blocks for the recovery termination manager (RTM): • **RTM2WA** associated with all TCBs in the dumped address space • **RTM2WA** associated with the TCB for the dump requester		 S	S
Cross memory status record and, if the dump requester held a cross memory local (CML) lock, the address of the ASCB for the address space whose local lock is held	S	S	
Dump header, mapped by AMDDATA See *z/OS MVS Data Areas, Vol 1 (ABEP-DALT)* for the AMDDATA mapping.	S	S	S
Functional recovery routine (FRR) stack for the current processor	S		
Interrupt handler save area (IHSA) for the home address space or, if a CML is held, for the address space whose local lock is held	S	S	
Logical communication area (LCCA) for each active processor In dumps requested by AR-mode callers, the LCCA includes the AR mode control blocks	S	S	
Physical configuration communication area (PCCA) for each active processor	S	S	
Program call link stack elements (PCLINK) stack elements: • Pointed to by PSASEL • Pointed to by the XSB associated with the IHSA in the dump • Pointed to by the SSRB and XSB for the SRB-mode dump requester • Associated with the suspended unit of work	 S S	 S S S	
Prefixed save area (PSA) for each active processor	S	S	
Save areas of register contents		S	
SDWA associated with the failure of a system routine	S		

Table 2-3. Customizing SVC Dump Contents through Summary Dumps (continued)

Summary Dump Contents	Disabled Summary Dump	Suspend Summary Dump	Enabled Summary Dump
Storage: The storage ranges and ASIDs requested in parameters on the SDUMP or SDUMPX macro	S	S	S
Storage: 4 kilobytes before and 4 kilobytes after: • The address in the program status word (PSW) • All valid unique addresses in the registers saved in the IHSA shown in the dump • All valid unique addresses in the registers saved in the SDWA shown in the dump • Instruction counter values of the external old PSW, program check old PSW, I/O old PSW, and restart old PSW saved in the PSAs of each active processor	S		
Storage: 4 kilobytes before and 4 kilobytes after: • All valid unique addresses in the registers saved in the SDWA shown in the dump • All valid unique addresses in the registers in the dump requester's register save area • All valid unique addresses in the PSWs in all SDWAs shown in the dump		S	
Storage: 4 kilobytes before and 4 kilobytes after: • All valid unique addresses in the PSWs in the RTM2WAs shown in the dump • All valid unique addresses in the registers in the RTM2WAs shown in the dump			S
Storage: When a PSWREGS parameter is specified on the SDUMP or SDUMPX macro, 4 kilobytes before and 4 kilobytes after: • The address in the PSW, if supplied in the PSWREGS parameter • The address in the general purpose registers, if supplied in the PSWREGS parameter The storage dumped is from the primary and secondary address spaces of the program issuing the SDUMP or SDUMPX macro. The control registers, if supplied in the PSWREGS parameter, are used to determine the primary and second address spaces. If access registers are also provided and the PSW indicates AR ASC mode, the access registers will also be used to locate the data.	S	S	S
Supervisor control blocks: • Current linkage stack • Primary address space number (PASN) access list • Work unit access list	S	S	S
Vector Facility control blocks: Global, CPU, and local work/save area vector tables (WSAVTG, WSAVTC, and WSAVTL) and work/save areas pointed to by addresses in the tables	S		
XSB associated with the IHSA in the dump	S	S	

References

See the following for information about control blocks listed in the above table:
- *z/OS MVS Data Areas, Vol 1 (ABEP-DALT)*
- *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)*
- *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*
- *z/OS MVS Data Areas, Vol 4 (RD-SRRA)*
- *z/OS MVS Data Areas, Vol 5 (SSAG-XTLST)*

Customizing Contents Through Operator Commands

The dump options list for SVC dumps can be customized through a DUMP operator command by all the ways shown in Table 2-4 on page 2-30.

Note: The contents of SVC dumps requested by SLIP operator commands are controlled only by the SLIP operator command. They are not affected by the IEACMD00 parmlib member or the CHNGDUMP command.

Nucleus Areas in Dumps

Dump options control the parts of the nucleus that appear in a dump. A diagnostician seldom needs to analyze all the nucleus. An installation can eliminate nucleus areas from dumps. If the IBM-supplied defaults are used:

- SVC dump for a SLIP operator command with ACTION=SVCD contains the read/write DAT-on nucleus
- SVC dump for an SDUMP or SDUMPX macro contains the nucleus map and certain control blocks

If no nucleus changes have been made, an installation should obtain one copy of the DAT-off nucleus to use with all dumps. To obtain this nucleus, enter a DUMP operator command with SDATA=ALLNUC and no other SDATA options. The nucleus does not change from one IPL to another, so one dump can be used again and again.

DAT, dynamic address translation, is the hardware feature that enables virtual storage. In the DAT-on part of the nucleus, the addresses are in virtual storage; in the DAT-off part of the nucleus, the addresses are in central storage.

Table 2-4. Customizing SVC Dump Contents through Operator Commands

Customization	Effect	Example
Use SDATA=NODEFAULTS on SDUMP or SDUMPX macro	**Change occurs:** At dump request **What changes:** Excludes the following SDATA default options currently in effect: • ALLPSA • SQA • SUMDUMP • IO • From all CHNGDUMP operator commands entered through the IEACMD00 parmlib member or through the console Exclusion is only for the dump being requested. Note that certain default system areas are not excluded; these areas are required for IPCS dump analysis. The CHNGDUMP operator command can override the NODEFAULTS option.	To minimize the amount of default data in the dump, code in the program: `SDUMPX SDATA=NODEFAULTS`
Replacing CHNGDUMP operator command in IEACMD00 parmlib member	**Change occurs**: At system initialization **What changes**: This command establishes the IBM-supplied dump options for SVC dumps for SDUMP or SDUMPX macros and DUMP operator commands; see "Contents of SVC Dumps" on page 2-21 for the list.	To add the link pack area (LPA) to all SVC dumps for SDUMP or SDUMPX macros and DUMP operator commands, while keeping the local system queue area (LSQA) and trace data, add the following command to IEACMD00: `CHNGDUMP SET,SDUMP=(LPA)`

Table 2-4. Customizing SVC Dump Contents through Operator Commands (continued)

Customization	Effect	Example

Entering CHNGDUMP operator command with SDUMP parameter on a console with master authority	**Change occurs**: Immediately when command is processed
	
	What changes:
	
	For the ADD mode: CHNGDUMP options are added to the current SVC dump options list and to any options specified in the macro or operator command that requested the dump. The options are added to all SVC dumps for SDUMP or SDUMPX macros and DUMP operator commands until another CHNGDUMP SDUMPX operator command is entered.
	
	For the OVER mode: CHNGDUMP options are added to the current SVC dump options list. The system ignores any options specified in the macro or operator command that requested the dump. The options override all SVC dumps for SDUMP or SDUMPX macros and DUMP operator commands until a CHNGDUMP SDUMP,ADD operator command is entered.
	
	For the DEL option: CHNGDUMP options are deleted from the SVC dump options list.
	
	When more than one CHNGDUMP operator command with SDUMPX is entered, the effect is cumulative.
		To add the LPA to all SVC dumps for SDUMP or SDUMPX macros and DUMP operator commands until changed by another CHNGDUMP SDUMP, enter:
		`CHNGDUMP SET,SDUMP=(LPA)`
		
		To add the CHNGDUMP SDUMPX options list to all SVC dumps:
		`CHNGDUMP SET,SDUMP,ADD`
		
		To override all SVC dumps with the CHNGDUMP SDUMPX options list:
		`CHNGDUMP SET,SDUMP,OVER`
		
		To remove LPA from the SDUMPX options list:
		`CHNGDUMP DEL,SDUMP=(LPA)`

Using an operator command parameter.
 	
Parameters on the DUMP operator command specify the contents for the dump being requested.
	Change occurs: At dump request
	
	What changes: The DUMP operator command parameter options are added to the dump options list, but only for the dump being requested.
		To add ALLNUC to this SVC dump, enter:
		`DUMP COMM=(MYDUMP1 5-9-88)`
		
		The system issues a message:
		`* 23 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND`
		
		Enter in reply:
		`REPLY 23,JOBNAME=MYJOB1, SDATA=(ALLNUC),END`

Tailoring SVC Dumps

Sometimes servers contain client-related data in address and dataspaces other than the client's which does not appear in the dump. In such cases, you can modify the contents of an SVC dump to provide additional problem determination data by creating a tailored SVC dump exit. In this manner you can specify a dump request without having to be responsible for identifying related server address and dataspaces and storage areas, all of which may be unknown and dynamic in nature.

You can create these SVC dump exits without modifying module IEAVTSXT. Use the CSVDYNEX macro to identify the component exit load module and associate it with the IEASDUMP.SERVER resource. The exit scans the current dump request and determines if data should be added to the dump. The data is added to the dump by identifying it in the appropriate SDMSE_OUTPUT area.

The tailored SVC dump exits are not called in any particular order. To ensure that the current requests are presented to an exit, the dump requests are updated between exit invocations. If an exit adds data to the dump, every exit is re-invoked until no additional changes are made. Because of the additional processing required, tailored SVC dump exits do not receive control by default. To cause the exit processing to take place, you must specify one of the following:

- SDATA=SERVERS in the DUMP command
- SDATA=SERVERS in the SDUMPX macro

Specifying SDATA=SERVERS on the CHNGDUMP command will cause this to become the local default.

Analyzing Summary SVC Dumps

The SUMDUMP or SUM option on the SDUMP or SDUMPX macro causes SVC dump to capture a summary dump. Two types of information are captured in summary dumps. First, index data for storage is captured. This index data can be formatted using the IPCS VERBX SUMDUMP command. The second type of information captured is the storage itself. Storage captured by summary dump processing can be viewed using IPCS by specifying the SUMDUMP option (for example, list.00003000.sumdump). IBM strongly recommends that you view the SUMDUMP output prior to investigating the usual portions of the dump. The SUMDUMP option provides different output to SDUMPX branch entries and SVC entries to SDUMP. For example, data included for branch entries to SDUMPX include PSA, LCCA, and PCCA control blocks, and data recorded for SVC entries to SDUMPX include RTM2WA control blocks. Each summary dump index record, when formatted using the IPCS VERBX SUMDUMP command, is displayed as “----tttt---- range-start range-end range-asid range-attributes”.

Example: Format of IPCS VERBX SUMDUMP Command

The following is an example format using the IPCS VERBX SUMDUMP command.

```
STORAGE TYPE       RANGE START       RANGE END      ASID ATTRIBUTES
REGISTER AREA--          0135F000          01363FFF 001E (COMMON)
REGISTER AREA-- 00000001_7F5AD000 00000001_7F5B0FFF 001E
```

The summary dump is formatted by the IPCS VERBEXIT SUMDUMP subcommand and has an index which describes what the summary contains. Summary dumps are not created for dumps taken with the DUMP command. Only dumps created by the SDUMP or SDUMPX macro contain summary dumps.

Note: During SVC dump processing, the system sets some tasks in the requested address space non-dispatchable; non-dispatchable tasks in the dump may have been dispatchable at the time of the problem.

Example: m IPCS VERBX SUMDUMP Command

The following is a partial example of a summary dump using the IPCS VERBX SUMDUMP command.

```
STORAGE TYPE         RANGE START         RANGE END      ASID ATTRIBUTES
                          023BCD70            023BCD7F 001E (COMMON)
SUMLSTA RANGE --          017E8000            017E8FFF 0001 (COMMON)
SUMLSTA RANGE --          01F9B000            01F9CFFF 0001 (COMMON)
SUMLSTA RANGE --          02166000            02167FFF 0001 (COMMON)
PSA ------------          00000000            00001FFF 001E (COMMON)
PCCA -----------          00F43008            00F4324F 001E (COMMON)
LCCA -----------          00F82000            00F82A47 001E (COMMON)
LCCX -----------          021C7000            021C771F 001E (COMMON)
INT HANDLER DUCT          02232FC0            02232FFF 001E (COMMON)
I.H. LINKAGE STK          02262000            0226202F 001E (COMMON)
REGISTER AREA --          0000E000            00010FFF 001E
REGISTER AREA --          00FC4000            00FC6FFF 001E (COMMON)
REGISTER AREA --  00000001_7F5AD000   00000001_7F5B0FFF 001E
REGISTER AREA --          7FFFE000            7FFFEFFF 001E
```

To examine the storage shown above, invoke the list command as follows:

```
IPCS LIST 00FC4000. SUMDUMP LEN(256) DISPLAY

****************************** TOP OF DATA ***********************************


LIST 00FC4000 ASID(X'001E') SUMDUMP LENGTH(X'0100')
AREA
ASID(X'001E') SUMDUMP ADDRESS(FC4000.) KEY(00)
00FC4000. 7F6BFFD0 7F6BFFD0 02259010 00000000 |",.}",.}........|
00FC4010. 0225D000 02259010 00000004 00000001 |..}.............|
00FC4020. 00000000 00FC3E10 00000000 00000000 |................|
00FC4030. 00000000 00000000 00000000 00800000 |................|
00FC4040. 06102000 00000000 00000000 00000000 |................|
00FC4050 LENGTH(X'10')==>All bytes contain X'00'
00FC4060. 00000000 02247CB8 00000000 00000000 |......@.........|
00FC4070. 02258040 0000164E 00008000 00000000 |... ...+........|
00FC4080 LENGTH(X'80')==>All bytes contain X'00'
****************************** END OF DATA ***********************************
```

Reference

See the SMDLR and SMDXR control blocks in *z/OS MVS Data Areas, Vol 4 (RD-SRRA)* for the record id values.

SUMDUMP Output For SVC-Entry SDUMPX

For an SVC entry, the storage captured in a summary dump can contain information that is not available in the remainder of the SVC dump if options such as region, LSQA, nucleus, and LPA were not specified in the dump parameters.

For each address space dumped, a summary dump index record is written with the ASID, plus the jobname and stepname for the last task created in the address space. The SUMDUMP output contains RTM2 work areas for tasks in address spaces that are dumped. Many of the fields in the RTM2WA provide valuable debugging information.

The summary dump data is dumped in the following sequence:

1. The ASID record is dumped for the address space.
2. The SUMLIST/SUMLSTA/SUMLSTL/SUMLST64 ranges and the PSWREGS, parameter list and data ID, data are dumped next. These contain information that is helpful in debugging the problem, and should be examined carefully.
3. All RTM2 work areas pointed to by all TCBs in this address space are dumped.
4. An address range table is built containing the following ranges, pointed to by the RTM2WA:
   - 4K before and after the PSW at the time of error (RTM2NXT1)
   - 4K before and after each register at the time of error (RTM2EREG).

   Duplicate storage is eliminated from this address range table to reduce the amount of storage dumped.

SUMDUMP Output for Branch-Entry SDUMPX

For branch entry to SDUMP, there are two types of summary dumps:

- Disabled summary dump - which performs the summary dump with the system disabled for interruptions. This means that all data to be dumped must be paged in at the time of the summary dump.
- Suspend summary dump - which is taken in two parts. The first part is similar to the disabled summary dump and dumps some of the global system control blocks. The second part runs with the system enabled for interruptions. This allows data to be dumped that is currently paged out, but was going to be modified by the recovery routine that requested SVC dump processing.

The SUMDUMP output for a branch entry to SVC dump might not match the data that is at the same address in the remainder of the dump. The reason for this is that SUMDUMP is taken at the entry to SVC dump while the processor is disabled for interruptions. The system data in the remainder of the dump is often changed because other system activity occurs before the dump is complete. The SUMDUMP output follows a header that contains the ASID of the address space from which the data was obtained.

The following conditions can occur that prevent SDUMPX from taking a disabled or suspend summary dump.

- The system is not able to obtain the necessary locks to serialize the real storage buffer (RSB).
- The system is in the process of modifying the storage queues and cannot satisfy the request for a RSB.
- No frames are available for a RSB.
- SVC dump encounters an error while holding serialization for the RSB.
- A critical frame shortage causes the system to steal the pages of the RSB.
- The SVC dump timer disabled interruption exit determines that SVC dump has failed and frees the RSB.

Analyzing Disabled Summary Dumps

For **disabled summary dumps**, records are dumped in the following order:

1. If a suspend summary dump was requested but could not be taken, the system attempts to obtain a disabled summary dump. If this occurs, an error record is written to that effect. If the system is unable to obtain a suspend summary dump and a disabled summary dump, then no summary data is available for the dump.

2. The XMEM ASID record is written that gives the ASID that is home, primary, secondary, and CML (if the CML lock is held).
3. The SUMLIST/SUMLSTA/SUMLSTL/SUMLST64 address ranges and the PSWREGS data are dumped.
4. The PSA, PCCA, LCCA, and LCCX for each processor are dumped.
5. The current PCLINK stack (pointed to by PSASEL) is dumped (if it exists).
6. If this is a SLIP request for a dump (ACTION=SVCD), then the SLIP reg/PSW area (pointed to by the SUMDUMP parameter list SDURGPSA) is dumped.

   The following address ranges are added to the address range table:
   - 4K before and after the PSW address at the time of the SLIP trap.
   - 4K before and after each address in the registers at the time of the SLIP trap.

   Duplicate storage is eliminated from this address range table to reduce the amount of data written to the dump data set.

   Note that if the primary and secondary ASIDs are different, the above address ranges are added to the table for both ASIDs.
7. The IHSA is dumped along with its associated XSB and PCLINK stack. The PSW and register addresses from the IHSA are added to the range table. This causes 4K of storage to be dumped around each address.
8. The caller's SDWA is dumped, if one exists. The PSW and register addresses from the SDWA are added to the range table. This causes 4K of storage to be dumped around each address.
9. The addresses in the address range table are dumped.
10. The super FRR stacks are dumped.
11. The global, local, and CPU work save area (WSA) vector tables are dumped. The save areas pointed to by each of these WSA vector tables are also dumped.
12. 4K of storage on either side of the address portion of the I/O old PSW, the program check old PSW, the external old PSW, and the restart old PSW saved in the PSA for all processors, are dumped.

Analyzing Suspend Summary Dumps

For **suspend summary dumps**, records are dumped in the following order:

1. The ASID: the PSA, PCCA, LCCA records, the IHSA, XSB, and the PCLINK stack, are all dumped with the system disabled in the same way they are dumped in steps 2, 4, and 5 for the disabled summary dump.

   At this point, an SRB is scheduled to the DUMPSRV address space and the current unit of work (SDUMP's caller) is suspended by using the STOP service. Data dumped at this point does not have to be paged in because the system is enabled. Cross memory functions are used to gain access to data in the caller's address space.
2. The SUMLIST/SUMLSTA/SUMLSTL/SUMLST64 address ranges and the PSWREGS data are dumped.
3. The caller's ASCB is dumped.
4. The suspended unit of work (SVC dump's caller) is dumped. This is either a TCB or an SSRB. The related PCLINK stacks are also dumped.
5. For TCB mode callers, the caller's SDWA is dumped. The PSW and register addresses from the SDWA are added to the range table. This causes 4K of

storage to be dumped around each address. All RTM2 work areas pointed to by this TCB and any associated SDWAs are all dumped.

For SRB mode callers, the SDWA is dumped. The PSW and register addresses from the SDWA are added to the range table. This causes 4K of storage to be dumped around each address. Also, the caller's register save area is added to the range table and the storage dumped.

Duplicate storage is eliminated from the address range table to reduce the amount of storage dumped.

6. After all the storage is saved in a virtual buffer in the DUMPSRV address space, the caller's unit of work is reset by using the RESET service. This allows SVC dump to complete and return to the caller. When SVC dump processing completes in the address space to be dumped, whatever processing was taking place in that address space when it was interrupted by SVC dump resumes. The rest of the dump is then scheduled from the DUMPSRV address space.

Analyzing an SVC Dump

This section shows you how to use IPCS to analyze an SVC dump. You would analyze an SVC dump because of one of the following:

- Dump output from the IPCS STATUS FAILDATA subcommand did not contain data for the abend being diagnosed.
- The problem involved multiple abends.
- The dump was taken but does not contain abend-related information.

This section contains the following topics, which, if followed in order, represent the procedure for analyzing an SVC dump:

- "Formatting the SVC Dump Header" on page 2-37
- "Looking at the Dump Title" on page 2-38
- "Displaying the Incident Token, Time and Type of Dump" on page 2-39
- "Locating Error Information" on page 2-40
- "Analyze TCB Structure" on page 2-43
- "Examining the LOGREC Buffer" on page 2-44
- "Examining the System Trace" on page 2-47
- "Looking at the Registers" on page 2-47
- "Other Useful Reports for SVC Dump Analysis" on page 2-49
- "Reading the SDUMPX 4K SQA Buffer" on page 2-50

Specifying the Source of the Dump

The first step in analyzing the dump is to specify the source of the dump that IPCS should format. In the IPCS dialog choose option 0 (DEFAULTS) and specify the name of the SVC dump data set on the "Source" line.

```
------------------------- IPCS Default Values ---------------------------------
COMMAND ===> 0


  You may change any of the defaults listed below.
  If you change the Source default, IPCS will display the current default
  Address Space for the new source and will ignore any data entered in
  the Address Space field.

  Source  ==> DSN('D46IPCS.SVC.CSVLLA.DUMP002')
  Address Space   ==>                        Ignored if Source is changed.
  Message Routing ==> NOPRINT TERMINAL
  Message Control ==> FLAG(WARNING)
  Display Content ==> NOMACHINE REMARK REQUEST NOSTORAGE SYMBOL

Press ENTER to update defaults.
Use the END command to exit without an update.
```

Press Enter to register the new default source name. Then press PF3 to exit the panel.

You can also use the SETDEF subcommand to specify the source. For the dump in the preceding example, enter:

```
SETDEF DSNAME('D46IPCS.SVC.CSVLLA.DUMP002')
```

IPCS does not initialize the dump until you enter the first subcommand or IPCS dialog option that performs formatting or analysis. At that time IPCS issues message BLS18160D to ask you if summary dump data can be used by IPCS. The summary dump data should always be used for an SVC dump because it is the data captured closest to the time of the failure. If you do not allow IPCS to use summary dump data, other data captured later for the same locations will be displayed, if available. Such data is less likely to be representative of the actual data at these storage locations at the time of the failure.

Formatting the SVC Dump Header

The SVC dump header contains the following information:

- SDWA or SLIP data
- Dump title, error identifier, and time of the dump
- Requestor of dump

This information describes the type of SVC dump and can tell you if the dump is a CONSOLE dump or a dump caused by the SLIP command. You would analyze these dumps differently.

Format data in the header of an SVC dump using the following IPCS subcommands:

LIST TITLE
STATUS FAILDATA
STATUS REGISTERS
STATUS WORKSHEET

The following sections give examples of how to use these IPCS subcommands (or IPCS dialog options, where applicable) to obtain the desired information.

Looking at the Dump Title

The dump title tells you the component name, component identifier and module name. You can find the dump title using the following IPCS subcommands:

LIST TITLE
STATUS WORKSHEET

You can also obtain the STATUS WORKSHEET report through option 2.3 of the IPCS dialog. First choose option 2 (ANALYSIS) from the primary option menu:

```
-------------------- z/OS 01.02.00 IPCS PRIMARY OPTION MENU ----------------
OPTION ===> 2

                                                      ********************
 0DEFAULTS  - Specify default dump and options      * USERID   - IPCSU1
 1BROWSE    - Browse dump data set                  * DATE     - 84/06/08
 2ANALYSIS  - Analyze dump contents                 * JULIAN   - 84.160
 3SUBMIT    - Submit problem analysis job to batch  * TIME     - 16:43
 4COMMAND   - Enter subcommand, CLIST or REXX exec  * PREFIX   - IPCSU1
 5UTILITY   - Perform utility functions             * TERMINAL - 3278
 6DUMPS     - Manage dump inventory                 * PF KEYS  - 24
 TTUTORIAL  - Learn how to use the IPCS dialog      ********************
 XEXIT      - Terminate using log and list defaults

Enter END command to end the IPCS dialog.
```

Then choose option 3 (WORKSHEET) from the analysis of dump contents menu:

```
-------------------- IPCS MVS ANALYSIS OF DUMP CONTENTS ---------------------
OPTION  ===> 3
To display information, specify the corresponding option number.

 1SYMPTOMS    - Symptoms                              *****************
 2STATUS      - System environment summary            * USERID  - IPCSU1
 3WORKSHEET   - System environment worksheet          * DATE    - 84/06/08
 4SUMMARY     - Address spaces and tasks              * JULIAN  - 84.160
 5CONTENTION  - Resource contention                   * TIME    - 16:44
 6COMPONENT   - MVS component data                    * PREFIX  - IPCSU1
 7TRACE       - Trace formatting                      * TERMINAL- 3278
 8STRDATA     - Coupling Facility structure data      * PF KEYS - 24
                                                        ******************
Enter END command to terminate MVS dump analysis.
```

IPCS displays a new panel with information similar to that in Figure 2-2 on page 2-39. The dump title is labelled at the top of the STATUS WORKSHEET report. The dump title is “Compon=Program Manager Library-Lookaside, Compid=SC1CJ, Issuer=CSVLLBLD.”

Reference

See *z/OS MVS Diagnosis: Reference* for an explanation of dump titles.

```
IPCS OUTPUT STREAM ------------------------------------------ LINE 0 COL
COMMAND ===>                                                  SCROLL ===
 ***************************** TOP OF DATA ****************************


                             MVS Diagnostic Worksheet

Dump Title: COMPON=PROGRAM MANAGER LIBRARY-LOOKASIDE,COMPID=SC1CJ,
             ISSUER=CSVLLBLD

CPU Model 2064 Version FF Serial no. 131512 Address 01
Date: 02/15/2001   Time: 20:33:34.680912  Local

Original dump dataset: SYS1.DUMP06

Information at time of entry to SVCDUMP:

HASID 001B  PASID 001B  SASID 001B  PSW 070C1000 8001AE5A

CML ASCB address 00000000  Trace Table Control Header address 7FFE3000
Dump ID: 001
Error ID: Seq 00019  CPU 0041  ASID X'001E'  Time 20:33:33.5
```

Figure 2-2. STATUS WORKSHEET Subcommand Sample Output — Dump Title

STATUS WORKSHEET also displays the error ID. In Figure 2-2, the dump ID is 001, error ID is sequence number 00051, ASID=X'001B', and processor 0000. Use this dump ID to match messages in SYSLOG and LOGREC records to the dump.

Displaying the Incident Token, Time and Type of Dump

The IPCS subcommand STATUS SYSTEM identifies

- The time of the dump
- The program requesting the dump
- An incident token that associates one or more SVC dumps requested for a problem on a single system or on several systems in a sysplex

The IPCS dialog does not have a menu option for STATUS SYSTEM. Instead you must enter the subcommand.

Figure 2-3 on page 2-40 is an example of a STATUS SYSTEM report. For a scheduled SVC dump, the following identifies the dump:

```
Program Producing Dump: SVCDUMP
Program Requesting Dump: IEAVTSDT
```

A dump requested by a SLIP or DUMP operator command is always a scheduled SVC dump.

For a synchronous SVC dump, the following identifies the dump:

```
Program Producing Dump: SVCDUMP
Program Requesting Dump: cccccccc
```

Where cccccccc is one of the following:

- The name of the program running when the system detected the problem
- SVCDUMP, if the system could not determine the failing task

SVC Dump

A SYSMDUMP ABEND dump is always a synchronous SVC dump.

```
SYSTEM STATUS:
  Nucleus member name: IEANUC01
   I/O configuration data:
     IODF data set name: SYS0.IODF43
     IODF configuration ID: CONGIG00
     EDT ID: 00
  Sysplex name: SYSPL1
  TIME OF DAY CLOCK: B566EA85 A0750707 02/15/2001 20:33:34.680912 local
  TIME OF DAY CLOCK: B567202A 89750707 02/16/2001 00:33:34.680912 GMT
  Program Producing Dump: SVCDUMP
  Program Requesting Dump: IEAVTSDT
  Incident token: SYSPL1   S4       06/23/1993 12:43:54.697367 GMT
```

Figure 2-3. Sample Output from the STATUS SYSTEM Subcommand

SYSTEM STATUS for an SVC dump contains an incident token. The request for the dump specifies the incident token or the system requesting the dumps provides it. The incident token consists of:

- The name of the sysplex
- The name of the system requesting the multiple dumps
- The date in Greenwich Mean Time (GMT)
- The time in GMT

Locating Error Information

Use the IPCS subcommand STATUS FAILDATA to locate the specific instruction that failed and to format all the data in an SVC dump related to the software failure. This report gives information about the CSECT involved in the failure, the component identifier, and the PSW address at the time of the error.

Note: For SLIP dumps or CONSOLE dumps, use SUMMARY FORMAT or VERBEXIT LOGDATA instead of STATUS FAILDATA.

Choose option 4 (COMMAND) from the IPCS primary option menu and enter the following command:

```
------------------------- IPCS Subcommand Entry -------------------------------
Enter a free-form IPCS subcommand, CLIST, or REXX exec invocation below:

===> STATUS FAILDATA
```

Use the PF keys to scroll up and down through the report. The following sections describe parts of the report.

Identifying the Abend and Reason Codes

Under the heading “SEARCH ARGUMENT ABSTRACT”, you will find the abend code and, if provided, an abend reason code.

```
:
 SEARCH ARGUMENT ABSTRACT

  PIDS/5752SC1CJ RIDS/CSVLLCRE#L RIDS/CSVLLBLD AB/S0FF0 REGS/09560 REGS/ 06026
   RIDS/CSVLLBLD#R

   Symptom             Description
   -------             -----------
   PIDS/5752SC1CJ      Program id: 5752SC1CJ
   RIDS/CSVLLCRE#L     Load module name: CSVLLCRE
   RIDS/CSVLLBLD       Csect name: CSVLLBLD
   AB/S0FF0            System abend code: 0FF0
   REGS/09560          Register/PSW difference for R09: 560
   REGS/06026          Register/PSW difference for R06: 026
   RIDS/CSVLLBLD#R     Recovery routine csect name: CSVLLBLD
:
```

Figure 2-4. Search Argument Abstract in the STATUS FAILDATA Report

In Figure 2-4, the abend code is X'FF0' with no reason code. See *z/OS MVS System Codes* for a description of the abend code and reason code.

The following IPCS reports also provide the abend and reason codes:
- VERBEXIT LOGDATA
- STATUS WORKSHEET
- VERBEXIT SYMPTOMS

Finding the System Mode

Below the “SEARCH ARGUMENT ABSTRACT“ section is information describing the system mode at the time of the error.

```
:

  Home ASID: 001B    Primary ASID: 001B    Secondary ASID: 001B
  PKM: 8000          AX: 0001              EAX: 0000

  RTM was entered because a task requested ABEND via SVC 13.
  The error occurred while: an SRB was in control.
  No locks were held.
  No super bits were set.
:
```

Figure 2-5. System Mode Information in the STATUS FAILDATA Report

The line that starts with “The error occurred...” tells you if the failure occurred in an SRB or TCB. In the example in Figure 2-5, the error occurred while an SRB was in control, which means you need to look under the heading SEARCH ARGUMENT ABSTRACT (see Figure 2-4) to find the CSECT and load module names. This is the module in which the abend occurred.

If an SRB service routine was in control, look under the heading SEARCH ARGUMENT ABSTRACT for the CSECT and load module names. This is the failing module.

In output from a SUMMARY FORMAT subcommand, look for the RB for the abending program. The RB has an RTPSW1 field that is nonzero.

In a dump requested by a SLIP operator command, use a STATUS CPU REGISTERS subcommand to see data from the time of the problem.

If the error had occurred while a TCB was in control, you would find the failing TCB by formatting the dump using the IPCS subcommand SUMMARY TCBERROR. See “Analyze TCB Structure” on page 2-43.

Identifying the Failing Instruction

The STATUS FAILDATA report also helps you find the exact instruction that failed. This report provides the PSW address at the time of the error and the failing instruction text. Note that the text on this screen is not always the failing instruction text. Sometimes the PSW points to the place where the dump was taken and not the place where the error occurred.

In Figure 2-6, the PSW at the time of the error is X'11E6A3C' and the instruction length is 4-bytes; therefore, the failing instruction address is X'11E6A38'. The failing instruction is 927670FB.

```
⋮
 OTHER SERVICEABILITY INFORMATION

   Recovery Routine Label:  CSVLEBLD
   Date Assembled:          00245
   Module Level:            HBB7705
   Subfunction:             LIBRARY-LOOKASIDE

 Time of Error Information

   PSW: 070C0000 811E6A3C   Instruction length: 04   Interrupt code: 0004
   Failing instruction text: E0009276 70FB5030 70F8D7F7

   Registers 0-7
   GR: 0002A017 00FBE800 00000000 00000076  00000C60 00FBE600 0002A016
00000017
   AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
   Registers 8-15
   GR: 012221A8 811E69F8 00000001 30000000  00FD82C8 811CAD90 011E69D0 011E69D0
   AR: 00000000 00000000 00000000 00000000  00000000 00000000 00005F60 00000000
⋮
```

Figure 2-6. Time of Error Information in the STATUS FAILDATA Report

The failing instruction text displayed in this report is always 12 bytes, 6 bytes before and 6 bytes after the PSW address. In this example, the failing instruction, 927670FB, is an MVI of X'76' to the location specified by register 7 + X'FB'.

Register 7 at the time of the error, shown under **Registers 0-7** above, contained a X'00000017'. The attempted move was to storage location X'112'. The first 512 bytes of storage are hardware protected. Any software program that tries to store into that area without authorization will receive a protection exception error and a storage protection exception error.

Reference

See *z/Architecture Principles of Operation* for information about machine language operation codes, operands, and interruption codes.

To find the module that abnormally terminated and the offset to the failing instruction, use the WHERE command. WHERE can identify the module or CSECT that the failing PSW points to.

Analyze TCB Structure

If a TCB was in control at the time of the error, use the IPCS subcommand SUMMARY TCBERROR to look at the TCB information and find the failing component. SUMMARY TCBERROR summarizes the control blocks for the failing address space. (To see all the fields in the control blocks, use SUMMARY FORMAT.) Scan the completion codes (field CMP) for each TCB to find the correct TCB. This report displays RBs from newest to oldest.

Figure 2-7 is an example of SUMMARY TCBERROR output. In this example the TCB at address 008E9A18 has a completion code of X'0C1.' The error occurred under this TCB. Once you have identified the failing TCB, you can follow the RB chain to the failing program.

```
 :
     NAME..... IEFSD060  ENTPT.... 00DA6308

  PRB: 008FFED0
     WLIC..... 00020006  FLCE.... 00C534A0  OPSW..... 070C2000 00DC766A
     LINK..... 008FFA10

  CDE: 00C534A0
     NAME..... IEESB605  ENTPT.... 00DC7000

  TCB:  008FF0D0
     CMP...... 00000000  PKF...... 80        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 008FD210  NDSP..... 00002000
     JSCB..... 008FF4FC  BITS..... 00000000  DAR...... 00
     RTWA..... 00000000  FBYT1.... 00
     Task non-dispatchability flags from TCBFLGS4:
      Top RB is in a wait
     Task non-dispatchability flags from TCBFLGS5:
      Secondary non-dispatchability indicator
     Task non-dispatchability flags from TCBNDSP2:
      SVC Dump is executing for another task
  PRB: 008E9F20
     WLIC..... 00020001  FLCDE.... 00C4CA38  OPSW..... 070C1000  00DAC66E
     LINK..... 018FF0D0

  CDE: 00C4CA38
     NAME..... IEFIIC    ENTPT.... 00DA6000
```

Figure 2-7. An Example of the SUMMARY TCBERROR Report (Part 1 of 2)

```
  TCB: 008E9A18
     CMP...... 940C1000  PKF...... 80        LMP...... FF        DSP...... FF
     TSFLG.... 20        STAB..... 008FD180  NDSP..... 00000000
     JSCB..... 008FF33C  BITS..... 00000000  DAR...... 01
     RTWA..... 7FFE3090  FBYT1.... 08


SVRB: 008FD7A8
     WLIC..... 00020000  FLCDE.... 00000000  OPSW..... 070C1000  82569B38
     LINK..... 008FD638

  PRB: 008E9750
     WLIC..... 00020033  FLCDE.... 14000000  OPSW..... 070C1000  80CE9AEE
     LINK..... 008FD638

  SVRB: 008FD638
     WLIC..... 0002000C  FLCDE.... 00000000  OPSW..... 070C1000  825E9768
     LINK..... 008FD4C8

  SVRB: 008FD4C8
     WLIC..... 00020001  FLCDE.... 00000000  OPSW..... 070C0000  00C47D52
     LINK..... 008FD358

  SVRB: 008FD358
     WLIC..... 00020053  FLCDE.... 00000000  OPSW..... 075C0000  00D64EOC
     LINK..... 008FF4D8

  PRB: 008FF4D8
     WLIC..... 00020014  FLCDE.... 008FF3D8  OPSW..... 078D0000  00006EF2
     LINK..... 008E9A18

  CDE: 008FF3D8
     NAME..... SMFWT     ENTPT.... 00006EB0
```

Figure 2-7. An Example of the SUMMARY TCBERROR Report (Part 2 of 2)

In this example, the most current RB is the SVRB at address 008FD7A8. This is the SVC dump's RB. The ESTAE's RB is the PRB at 008E9750. The ESTAE issued an SVC 33. The RB for the recovery termination manager (RTM) is the SVRB at 008FD638. RTM issued an SVC C to attach the ESTAE. The X'0C1' abend occurred under the SVRB at 008FD4C8. The last interrupt was a 1 at the address indicated in the old PSW field (OPSW). The next RB in the chain shows an SVC X'53' (SMFWTM) had been issued. This is the code the X'0C1' occurred in.

For a scheduled dump, the abnormally terminating TCB can generally be found by scanning for a nonzero completion code. If there is no code, scan the system trace for the abend. The trace identifies the ASID number and TCB address for each entry. See "Examining the System Trace" on page 2-47.

Use the STATUS or the STATUS REGS subcommand to find the data set name and the module name of the SVC dump requester.

Examining the LOGREC Buffer

Use the IPCS subcommand VERBEXIT LOGDATA to view the LOGREC buffer in a dump. This report might repeat much of the information contained in the STATUS FAILDATA report, but it helps to identify occasions when multiple error events caused the software failure.

The example in Figure 2-8 on page 2-45 shows how multiple errors can appear in the LOGREC buffer. Abend X'0D5' is the first abend and X'058' is the second. Always check for multiple errors in the VERBEXIT LOGDATA report that are in the same address space or a related address space and are coincident with or precede the SVC dump.

```
 TYPE:  SOFTWARE RECORD      REPORT:  SOFTWARE EDIT REPORT           DAY.YEAR
        (PROGRAM INTERRUPT)                                REPORT DATE: 235.91
 SCP:   VS 2 REL 3                                          ERROR DATE: 126.91
                             MODEL:   3090                               HH:MM:SS.TH
                             SERIAL:  272804                      TIME: 13:27:59.86

 JOBNAME: LSCMSTR

ERRORID: SEQ=01196  CPU=0042  ASID=000C  TIME=13:27:59.6

 SEARCH ARGUMENT ABSTRACT

   PIDS/####SC1C5 RIDS/NUCLEUS#L RIDS/IEAVEDS0 AB/S00D5 PRCS/00000021 REGS/0F120
   RIDS/IEAVEDSR#R

   SYMPTOM             DESCRIPTION
   -------             -----------
   PIDS/####SC1C5      PROGRAM ID: ####SC1C5
   RIDS/NUCLEUS#L      LOAD MODULE NAME: NUCLEUS
   RIDS/IEAVEDS0       CSECT NAME: IEAVEDS0
   AB/S00D5            SYSTEM ABEND CODE: 00D5
   PRCS/00000021       ABEND REASON CODE: 00000021
   REGS/0F120          REGISTER/PSW DIFFERENCE FOR R0F: 120
   RIDS/IEAVEDSR#R     RECOVERY ROUTINE CSECT NAME: IEAVEDSR

 OTHER SERVICEABILITY INFORMATION

   RECOVERY ROUTINE LABEL:  IEAVEDSR
   DATE ASSEMBLED:          08/23/89
   MODULE LEVEL:             UY41669
   SUBFUNCTION:             DISPATCHER

 TIME OF ERROR INFORMATION

   PSW: 440C0000 80FEFC56   INSTRUCTION LENGTH: 04   INTERRUPT CODE: 0021
   FAILING INSTRUCTION TEXT: 1008B777 1008B225 000007FE
   TRANSLATION EXCEPTION IDENTIFICATION: 00000041
   REGISTERS 0-7
   GR: 00000041 00F9A0C0 00000000 00000000  00000000 008DE188 008E8C78 00000001
   REGISTERS 8-15
   GR: 00F97280 0103AB6A 00FF1B08 008DE188  0000000C 000C0041 80FF6510 00FEFB36

   HOME ASID: 000C    PRIMARY ASID: 000C    SECONDARY ASID: 000C
   PKM: 8000          AX: 0001

   RTM WAS ENTERED BECAUSE OF A PROGRAM CHECK INTERRUPT.
   THE ERROR OCCURRED WHILE A LOCKED OR DISABLED ROUTINE WAS IN CONTROL.
   NO LOCKS WERE HELD.
   SUPER BITS SET: PSADISP  - DISPATCHER
```

Figure 2-8. Sample Output from the VERBEXIT LOGDATA Subcommand (Part 1 of 2)

```
:
 TYPE:  SOFTWARE RECORD      REPORT:  SOFTWARE EDIT REPORT         DAY.YEAR
       (SVC 13)                                     REPORT DAT 235.91
 SCP:   VS 2 REL 3                                   ERROR DAT 126.91
                             MODEL:   3090                    HH:MM:SS.TH
                             SERIAL:  272804               TIM 13:27:59.94

 JOBNAME: LSCMSTR
 ERRORID: SEQ=01197  CPU=0000  ASID=000C  TIME=13:27:59.6

 SEARCH ARGUMENT ABSTRACT

   AB/S0058

   SYMPTOM            DESCRIPTION
   -------            -----------
   AB/S0058           SYSTEM ABEND CODE: 0058


SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE

  PROGRAM ID
  LOAD MODULE NAME
  CSECT NAME
  RECOVERY ROUTINE CSECT NAME
  RECOVERY ROUTINE LABEL
  DATE ASSEMBLED
  MODULE LEVEL
  SUBFUNCTION

TIME OF ERROR INFORMATION

  PSW: 470C8000 00FDC266   INSTRUCTION LENGTH: 02   INTERRUPT CODE: 000D
  FAILING INSTRUCTION TEXT: 00000000 0A0D0A06 00000000

  REGISTERS 0-7
  GR: 00A5D7A8 80058000 00000041 022DC780  008EDF00 008FBC7C 00F86A00 026E01F0
  REGISTERS 8-15
  GR: 026E0160 00000000 8001AC96 0001BC96  00000000 000188F0 8001B194 00000020

  HOME ASID: 000C    PRIMARY ASID: 000C    SECONDARY ASID: 000C
  PKM: 8000          AX: 0001

  RTM WAS ENTERED BECAUSE AN SVC WAS ISSUED IN AN IMPROPER MODE.
  THE ERROR OCCURRED WHILE AN ENABLED RB WAS IN CONTROL.
  NO LOCKS WERE HELD.
  NO SUPER BITS WERE SET.
```

Figure 2-8. Sample Output from the VERBEXIT LOGDATA Subcommand (Part 2 of 2)

When viewing the VERBEXIT LOGDATA report, skip the hardware records to view the software records. Search for the first software record.

The field “ERRORID=” gives the error identifier for the software failure. The error identifier consists of the sequence number, ASID, and time of the abend. By matching this identifier with error identifiers from other reports, you can tell if this is the same abend you have been analyzing or if it is a different abend. See “Interpreting Software Records” on page 14-19 for more information.

Examining the System Trace

The system trace table describes the events in the system leading up to the error. The trace table is helpful when the PSW does not point to the failing instruction, and to indicate what sequence of events preceded the abend.

IPCS option 2.7.4 formats the system trace. The report is long. IBM recommends scrolling to the end of the report, then backing up to find the trace entry for the abend. Type an M on the command line and press F8 to scroll to the bottom of the report.

After you find the entry for the abend, start at the PSW where the dump was taken and track the events in the table to find where the failing instruction is in the code.

The system trace report marks important or significant entries with an asterisk. In Figure 2-9 "*SVC D" in the "IDENT CD/D" column identifies the PSW where the program took the dump. Prior to the SVC D are three PGM (program check) entries. PGM 001 has an asterisk next to it, indicating that the program check was unresolved. The next entry, RCVY PROG, identifies a recovery program that failed because it issued the SVC D a few entries later. See Chapter 8, "System Trace" on page 8-1 to recognize significant entries in the system trace table.

```
PR  ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
                                                 UNIQUE-4 UNIQUE-5 UNIQUE-6
 :

 01 0094 00AF7D18   DSP       070C0000 81EA7000  00000000 00000000 0000800C
 01 0094 00AF7D18   SVC    78 070C0000 81EA7048  00000002 00000278 00000000
 01 0094 00AF7D18  SVCR    78 070C0000 81EA7048  00000000 00000278 03300D88
 01 0094 00AF7D18   PGM   010 070C0000 81EA704A  00040010 03300D8C
 01 0094 00AF7D18   PGM   011 070C0000 81EA704A  00040011 03300D8C
 01 0094 00AF7D18   SVC    77 070C2000 81EA7088  81EA7000 00000000 00050000
 01 0094 00AF7D18  SVCR    77 070C2000 81EA7088  00000000 00000000 40000000
 01 0094 00AF7D18  *PGM   001 070C0000 83300FAA  00020001 03300D8C
 01 0094 00AF7D18  *RCVY PROG                    940C1000 00000001 00000000
 02 0001 00000000   I/O   1A2 070E0000 00000000  0080000E 060246C0 0C000001
 02 0054 00AD7300   SRB       070C0000 810537E0  00000054 00F3C9F8 00F3CA40
 01 0054 00AF7D18   SSRV  12D          810B9CEE  00AF7D18 000C0000 00000000
 01 0094 00AF7D18   SSRV  12D          810B9D0E  00AF7D18 000B0000 00000000
 01 0094 00AF7D18   DSP       070C0000 810BF664  00000000 00000000 40000000
 01 0094 00AF7D18  *SVC     D 070C0000 810BF666  00000040 00000000 40000000
 01 0054 00000000  SSRV   10F          00000000  00F83E80 00AD7300 00AC5040
```

Figure 2-9. An Example of Output from the IPCS Subcommand SYSTRACE

Looking at the Registers

Use the IPCS subcommand STATUS REGISTERS to display the registers for the TCBs and RBs. SUMMARY REGS gives the same information in a different format.

This report identifies the PSW, ASID and register values just as the STATUS FAILDATA report, but STATUS REGISTERS also gives the control register values.

```
CPU STATUS:

 PSW=070C1000  80FE5CFC  (RUNNING IN PRIMARY, KEY 0,  AMODE 31, DAT ON)
     DISABLED FOR PER
   ASID(X'001B') 00FE5CFC. IEANUC01.IEAVESVC+05FC IN READ ONLY NUCLEUS
  ASCB27 at F3FA00, JOB(LLA), for the home ASID
  ASXB27 at 9FDF00 for the home ASID. No block is dispatched
  HOME ASID: 001B PRIMARY ASID: 001B SECONDARY ASID: 001B

  GPR VALUES
      0-3  80000000  80FF0000  009FF5A0  00FC4E88
      4-7  009F8E88  009FD358  80FE5CD6  00F3FA00
      8-11 00000000  80FE579C  009FD418  7FFFE2C0
     12-15 7FFE0000  00006730  00FE6200  80014910

  ACCESS REGISTER VALUES
      0-3  7FFEA5CC  00000000  00000000  00000000
      4-7  00000000  00000000  00000000  00000000
      8-11 00000000  00000000  00000000  00000000
     12-15 00000000  00000000  00005F60  8210532A

ALET TRANSLATION
AR 00    Not translatable
AR 14    Not translatable
AR 15    Not translatable

  CONTROL REGISTER VALUES
      0-3  5EB1EE40  00A2007F  007CCDC0  8000001B
      4-7  0001001B  00C506C0  FE000000  00A2007F
      8-11 00000000  00000000  00000000  00000000
     12-15 0082E07B  00A2007F  DF880C71  7FFE7008
```

Figure 2-10. Sample of the STATUS REGISTERS Report

The example output in Figure 2-10 shows the address in the PSW is X'0FE5CFC', the ASID is X'1B', and the failing instruction is located in offset X'5FC' in the CSECT IEAVESVC in the module IEANUC01 in the nucleus. You can now browse the dump at this location and look at the specific failing instruction. You could also use the information about the registers to find out more about the error if the address in the PSW does not point to the failing instruction.

This report identifies the PSW, ASID and register values just as the STATUS FAILDATA report, but STATUS REGISTERS also gives the control register values.

```
CPU STATUS:
 PSW=070C4000 00FC5C96
 (Running in AR, key 0, AMODE 24, DAT ON)
 DISABLED FOR PER
 ASID(X'001E') FC5C96. STRUCTURE(Cvt)+D6 IN READ/WRITE NUCLEUS
 ASID(X'001E') FC5C96. IEANUC01.IEAVCVT+0116 IN READ/WRITE NUCLEUS
 ASID(X'001E') FC5C96. STRUCTURE(Dcb)+0152 IN READ/WRITE NUCLEUS
 ASID(X'001E') FC5C96. STRUCTURE(Dcb)+015A IN READ/WRITE NUCLEUS
 ASCB30 at F90B80, JOB(ORANGE), for the home ASID
 ASXB30 at 6FDE90 and TCB30D at 6E7A68 for the home ASID
 HOME ASID: 001E PRIMARY ASID: 001E SECONDARY ASID: 001E

 General purpose register values
 0-1 00000000_00000020 00000000_84058000
 2-3 00000000_00000000 00000001_00004000
 4-5 00000000_01F9B9A8 00000000_01F9B9A8
 6-7 00000000_00000000 00000000_01F9BE10
 8-9 00000000_00000000 00000000_FFFFFFFC
 10-11 00000000_00000000 00000000_00FDAC58
 12-13 00000000_01560410 00000000_01F9BB08
 14-15 00000000_8155E5A8 00000000_0000003C

 Access register values
 0-3 00000000 00000000 00000000 00000000
 4-7 00000000 00000000 00000000 00000000
 8-11 00000000 00000000 00000000 00000000
 12-15 00000000 00000000 00000000 00000000

 Control register values
 Left halves of all registers contain zeros
 0-3 5F29EE40 0374C007 008D0A40 00C0001E
 4-7 0000001E 02A30780 FE000000 0374C007
 8-11 00020000 00000000 00000000 00000000
 12-15 0294EE43 0374C007 DF882A2F 7F5CD4B0
```

Figure 2-11. Sample of the STATUS REGISTERS Report Run in z/Architecture Mode

Other Useful Reports for SVC Dump Analysis

To collect further SVC dump data, use any of the following commands.

IPCS Subcommand	Information in the Report
STATUS CPU REGISTERS DATA CONTENTION	Data about the abend, current ASID, and task.
SUMMARY FORMAT	All fields in the TCBs and the current ASID.
TCBEXIT IEAVTFMT 21C.%	The current FRR stack.
LPAMAP	The entry points in the active LPA and PLPA.
VERBEXIT NUCMAP	A map of the modules in the nucleus when the dump was taken.
VERBEXIT SUMDUMP	The data dumped by the SUMDUMP option on the SDUMPX macro.
VERBEXIT MTRACE	The master trace table.
VERBEXIT SYMPTOMS	The primary and secondary symptoms if available.

Note: Use the VERBEXIT SYMPTOMS subcommand last in your SVC dump analysis. Other subcommands can add symptoms to the dump header record. This ensures VERBEXIT SYMPTOMS provides all symptoms available from the dump.

Reading the SDUMPX 4K SQA Buffer

The following SVC dumps contain problem data in an SDUMPX 4K system queue area (SQA) buffer:

- An SVC dump requested by a SLIP operator command
- Other SVC dumps, when indicated in the explanation of the dump title.
- An SVC dump requested by an SDUMP or SDUMPX macro with a BUFFER=YES parameter

To obtain the buffer, use the following IPCS subcommand:

```
LIST 0 DOMAIN(SDUMPBUFFER) LENGTH(4096)
```

This table describes the fields in the SQA buffer and should be used for diagnosis.

Offset	Length	Content
0(0)	4	The characters, TYPE
4(4)	4	RTM/SLIP processing environment indicator: X'00000001': RTM1 X'00000002': RTM2 X'00000003': MEMTERM X'00000004': PER
8(8)	4	The characters, CPU
12(C)	4	Logical processor identifier (CPUID)
16(10)	4	The characters, REGS
20(14)	64	General purpose registers 0 through 15 at the time of the event
84(54)	4	The characters, PSW
88(58)	8	The program status word (PSW) at the time of the event
96(60)	4	The characters, PASD
100(64)	2	The primary address space identifier (ASID) at the time of the event
102(66)	4	The characters, SASD
106(6A)	2	The secondary ASID at the time of the event
108(6C)	4	The characters, ARS
112(70)	64	Access registers 0 through 15 at the time of the event.
176(B0)	variable	One of the following, as indicated by the RTM/SLIP processing environment indicator at offset 4 of the buffer: • The system diagnostic work area (SDWA), if offset 4 is 1 (RTM1) • The recovery termination manager 2 (RTM2) work area (RTM2WA), if offset 4 is 2 (RTM2) • The address space control block (ASCB), if offset 4 is 3 (MEMTERM) • The PER interrupt code, if offset 4 is 4 (PER)

Chapter 3. Transaction Dump

Transaction dump is like the CHECK ENGINE light on the dashboard of your car when something's wrong.............it lets you know where the origin of the trouble is.

A Transaction dump provides a representation of the virtual storage for an address space when an error occurs. Typically, an application requests the dump from a recovery routine when an unexpected error occurs.

Transaction dumps come in the following types, depending on how they are requested.

- **Asynchronous Transaction Dump:**

  The requester's IEATDUMP macro invocation specifying ASYNC=YES issues an instruction to obtain the dump under the current task. Transaction dump creates a task under the current task to handle dump processing. The application returns control to the requester and is available once the request has been initiated.
- **Synchronous Transaction Dump:**

  The requester's IEATDUMP macro invocation issues an instruction to obtain the dump under the current task. The application returns control to the requester and is available once the dump data has been written into a dump data set.

Each Transaction dump also contains a summary dump, if requested. The summary dump supplies copies of selected data areas taken at the time of the request. Specifying a summary dump also provides a means of dumping many predefined data areas simply by specifying one option. This summary dump data is not mixed with the Transaction dump because in most cases it is chronologically out of step. Instead, each data area selected in the summary dump is separately formatted and identified. IBM recommends that you request summary dump data.

Major Topics

This chapter includes information system programmers need to know about Transaction dump and Transaction dump processing:

- "Using Pre-Allocated Dump Data Sets" on page 3-2
- "Using Automatically Allocated Dump Data Sets" on page 3-3
- "Obtaining Transaction Dumps" on page 3-5
- "Printing, Viewing, Copying, and Clearing a Pre-Allocated or a Dump Data Set" on page 3-6
- "Contents of Transaction Dumps" on page 3-7
- "Analyzing Summary Transaction Dumps" on page 3-13
- "Analyzing a Transaction Dump" on page 3-14

Reference

See *z/OS MVS Programming: Authorized Assembler Services Guide* for information any programmer needs to know about programming the IEATDUMP macro to obtain a Transaction Dump:

- Deciding when to request a Transaction dump
- Understanding the types of Transaction Dumps that MVS produces
- Designing an application program to handle a specific type of Transaction dump
- Identifying the data set to contain the dump
- Defining the contents of the dump

- Suppressing duplicate Transaction dumps using dump analysis and elimination (DAE)

Planning Data Sets for Transaction Dumps

Transaction dump processing stores data in dump data sets that you pre-allocate manually, or that are allocated automatically, as needed.

Using Pre-Allocated Dump Data Sets

To prepare your installation to receive Transaction dumps, you need to provide a dump data set. These data sets will hold the Transaction dump information for later review and analysis. This section describes how to set up the Transaction dump data sets, including:

- "Allocating Dump Data Sets With Secondary Extents"
- "Specifying Dump Data Sets" on page 3-3
- "Controlling Dump Data sets" on page 3-3

Allocating Dump Data Sets With Secondary Extents

Allocate a dump data set using the following requirements:

- Select a device with a track size of at least 4160 bytes. The system writes the dump in blocked records of 4160 bytes.
- Allocate the data set before requesting a dump. Allocation requirements are:
  - UNIT: A permanently resident volume on a direct access device.
  - DISP: Catalog the data set (CATLG). Do not specify SHR.
  - VOLUME: Allocating the dump data set on the same volume as the page data set could cause contention problems during dumping, as pages for the dumped address space are read from the page data set and written to the dump data set.
  - SPACE: An installation must consider the size of the page data set that will contain the dump data. The data set must be large enough to hold the amount of data as defined by VIO pages, and pageable private area pages.

    Transaction dump processing allows secondary extents to be specified when large dump data sets are too large for the amount of DASD previously allocated. An installation can protect itself against truncated dumps by specifying secondary extents and by leaving sufficient space on volumes to allow for the expansion of the dump data sets.

    For the SPACE keyword, you can specify CONTIG to make reading and writing the data set faster. Request enough space in the primary extent to hold the smallest Transaction dump expected. Request enough space in the secondary extent so that the primary plus the secondary extents can hold the largest Transaction Dump. The actual size of the dump depends on the dump options in effect when the system writes the dump.

    Estimate the largest dump size as follows:

```
Bytes of SDATA options + bytes in largest region size = Result1
Result1 * number of address spaces in dump            = Result2
PLPA * 20%                                            = Result3
Bytes of requested data space storage                 = Result4

Result2 + Result3 + Result4         = Bytes in Transaction dump
```

    Where:
    - `Result1`, `Result2`, `Result3`, `Result 4`: Intermediate results
    - `SDATA options`: Described in "Contents of Transaction Dumps" on page 3-7
    - `PLPA`: Pageable link pack area

For the size of the smallest dump, use the default options for the IEATDUMP macro. The difference between the largest dump and the smallest dump will be the size of the secondary extent.

Example: Calculating the Largest Amount of Storage

For example, to calculate the largest amount of storage required for a 3390 DASD, assume that, from the above calculations, the records needed for the Transaction dump amount to 6096 kilobytes. There are 12 records per track and 15 tracks per cylinder. To determine the number of cylinders needed to allocate a data set of this size, do the following:

- For 6096 kilobytes of storage, you will need space for 1524 Transaction dump records (6096 / 4 kilobytes per record).
- With 12 records per track, you will require 127 (1524 / 12 records) tracks.
- Therefore, the data set would require 9 cylinders (127 / 15 tracks per cylinder) for allocation.

Note: If you are not receiving the dump data you require, increase the size of the dump data set. You will receive system message IEA822I.

Reference

See *z/OS MVS Programming: Authorized Assembler Services Reference ENF-IXG* for information about the default dump options of the IEATDUMP macro.

Specifying Dump Data Sets

When planning a dump, remember that the data sets frequently contain sensitive data (user or installation confidential information, logon passwords, encryption keys, etc.). Protect these data sets with RACF to limit access to them.

The data sets are on direct access only. The direct access data set must be on a permanently resident volume; that is, the data set must be allocated. These dump data sets cannot be shared by more than one system.

All dump data sets should not be on the same pack. A pack should contain enough storage to allow the dump data sets to allocate secondary extent space, if needed.

Controlling Dump Data sets

After initialization, use a post dump exit routine to copy the dump to another data set. IEAVTSEL is a list of installation exit routines to be given control for SYSUDUMP or Transaction dump when dump processing ends.

Reference

See *z/OS MVS Installation Exits* for more information about the IEAVTSEL post dump exit name list.

Using Automatically Allocated Dump Data Sets

Transaction dump processing supports automatic allocation of dump data sets at the time the system writes the dump to DASD. The dump is allocated from the generic resource SYSALLDA. When the system writes a dump, it allocates a data set of the correct size. Automatic allocation is not multi-volume or striped. If automatic allocation fails, message IEA820I is issued, and the dump is deleted.

Naming Automatically Allocated Dump Data Sets

The application has control of the name of the data sets created by the automatic allocation function, and you can select a name-pattern to allow for dump data set organization according to your needs. The name is determined through an installation-supplied pattern on the DSN(AD) keyword in the IEATDUMP macro. A set of symbols is available so that you can include the following kinds of information in the names of your automatically allocated dump data sets:

- System name
- Sysplex name
- Job name
- Local and GMT time and date

You can specify a name-pattern to generate any name acceptable under normal MVS data set name standards. For a description of data set name standards, see the description of data set names in *z/OS DFSMS: Using Data Sets*.

Setting up Allocation Authority: To allocate dump data sets automatically, the caller's and/or DUMPSRV address space must have authority to allocate new data sets. Do the following:

1. **Associate the caller's and/or DUMPSRV address space with a user ID.**

   If you have RACF Version 2 Release 1 installed, use the STARTED general resource class to associate the caller or DUMPSRV with a user ID. For this step, the RACF started procedures table, ICHRIN03, must have a generic entry.

   If you have an earlier version of RACF, use the RACF started procedures table, ICHRIN03.
2. **Authorize caller's or DUMPSRV user ID to create new dump data sets using the naming convention in the following topic.**

   With the high-level qualifier of SYS1, the data sets are considered *group* data sets. You can assign CREATE group authority to the caller's user ID within that group.

References

See the following:

- *z/OS Security Server RACF System Programmer's Guide* for information about the RACF started procedures table.
- *z/OS Security Server RACF Security Administrator's Guide* for information on using the STARTED general resource class and on controlling creation of new data sets.

Establishing a Name Pattern: Establishing the name pattern for the dump data sets is accomplished by the DSN(AD) keyword. Names must conform to standard MVS data set naming conventions and are limited to 44 characters, including periods used as delimiters between qualifiers. To allow meaningful names for the dump data sets, several symbols are provided that are resolved when the dump data is captured in virtual storage. For a complete list of the symbols you can use, see the explanation of DUMPDS NAME= in *z/OS MVS System Commands*. For a description of data set name standards, see the description of data set names in *z/OS DFSMS: Using Data Sets*.

Note: The &SEQ. symbol is not supported for TDUMPs.

When determining the pattern for the dump data set names, consider any automation tools you may have at your installation that work on dump data sets.

The following describes a name pattern:

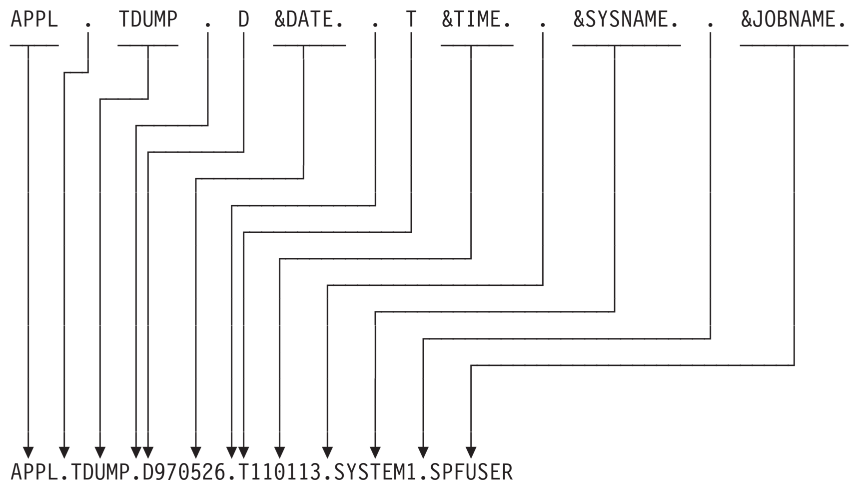

Figure 3-1. SPFUSER Name Pattern for Automatically Allocated Dump Data Set

Notice that the symbols are resolved into date and time, so they are preceded by an alphabetic character to conform to MVS data set name requirements. Also, the symbol starts with an ampersand (&) and ends with a period (.), resulting in a name pattern that has double periods when a symbol finishes a qualifier. One period ends the symbol, and the second serves as the delimiter between qualifiers of the generated data set name.

Automatic allocation of dump data sets is managed through the ISPF DSLIST on the high level qualifier (HLQ).

Communication from the System

The system communicates about automatic allocation of dump data sets using two messages:

- IEA822I is issued when a complete or partial dump is taken. IEA822I is an informational message, it will not be issued highlighted.
- IEA820I is issued once per Transaction dump when the dump cannot be taken or allocation fails. IEA820I is an informational message, it will not be issued highlighted.

Obtaining Transaction Dumps

Obtain a Transaction dump by issuing a IEATDUMP macro in an authorized or unauthorized program.

For dumps written to user-created dataspaces, the dataspace STOKEN and origin are identified on the IEATDUMP macro. Dumps to data sets are specified on the DCB or dump dataset name parameter of the IEATDUMP macro.

In a sysplex, you may need dumps from more than one address space to collect all of the problem data. These dumps need to be requested at the same time. To request these multiple dumps, issue a IEATDUMP macro with a REMOTE parameter specifying the other address spaces involved in the problem. To help you set up these requests, the parameter can contain wildcards. If the installation gives names that form patterns to the systems in the sysplex and to jobs for associated work, you can use wildcards, * and ?, to specify the names. For example, use the name TRANS? for the jobnames TRANS1, TRANS2, and TRANS3 and the name TRANS* for TRANS1, TRANS12, and TRANS123.

Note: If a Transaction dump uses the REMOTE parameter to dump one or more address spaces on a pre-release 4 system, the result will be a single SVC dump containing the requested data, instead of one or more Transaction dumps written to data set names specified with the DSN parameter. Issue the DISPLAY DUMP,STATUS command to determine the name of this SVC dump.

Issuing a Macro for Transaction dump

Use an IEATDUMP macro to request a Transaction Dump. The system writes the dump to a user-created dataspace, or a dump data set.

If the dump is written to a user-supplied Transaction dump data set, the program provides a data control block (DCB) for the data set, opens the DCB before issuing the IEATDUMP macro, and closes the DCB after the dump is written. For a synchronous dump, the close should occur when the system returns control to the requester. For an asynchronous dump, the close should occur when the ECB is posted.

If the dump is written to a user-created dataspace, the STOKEN and origin are identified on the IEATDUMP macro by the DSP_TOKEN and DSP_ORIGIN keywords. The caller also supplies the address of the output field DSP_RECORDS@ which is updated with the number of records written to the dataspace.

Example: Requesting a Synchronous Dump

To write a synchronous dump to a data set whose DCB address is in register 3 and whose header address is in register 4:

```
IEATDUMP DCBAD=(3), HDRAD=(4)
```

Reference

See *z/OS MVS Programming: Authorized Assembler Services Guide* for information about requesting an asynchronous Transaction dump and a synchronous Transaction Dump.

Printing, Viewing, Copying, and Clearing a Pre-Allocated or a Dump Data Set

Transaction Dumps are unformatted when created. Use IPCS to format a dump and then view it at a terminal or print it.

Example: JCL to Print, Copy, and Clear the Dump Data Set

For a pre-allocated data set or a dump data set, this JCL does the following:
- Uses the Transaction dump in the APPL.TDUMP00 data set. The IPCSTDMP DD statement identifies this data set.
- Deletes the IPCS dump directory in the DELETE(DDIR) statement. This statement uses the USERID of the batch job in the directory identification.
- Allocates the dump directory through the BLSCDDIR statement. The default is volume VSAM01. The example shows VSAM11. Override the default volume with the desired volume.
- Formats the dump using the IPCS subcommands in LIST 0. To use this example, replace the LIST 0 command with the desired IPCS subcommands or a CLIST. See *z/OS MVS IPCS User's Guide* for CLISTs.

```
//IPCSJOB  JOB
//IPCS     EXEC PGM=IKJEFT01,DYNAMNBR=75,REGION=1500K
//SYSPROC  DD   DSN=SYS1.SBLSCLI0,DISP=SHR
//IPCSTDMP DD   DSN=APPL.TDUMP00,DISP=SHR
//SYSUDUMP DD   SYSOUT=*
//IPCSTOC  DD   SYSOUT=*
//IPCSPRNT DD   SYSOUT=*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   *
DELETE(DDIR) PURGE CLUSTER
BLSCDDIR VOLUME(VSAM11)
IPCS NOPARM
SETDEF DD(IPCSTDMP) LIST NOCONFIRM
LIST 0
END
/*
```

Contents of Transaction Dumps

Transaction Dumps share parmlib member IEADMR00 to establish the dump options list at system initialization. The IBM-supplied IEADMR00 parmlib member specifies dump options NUC, SQA, LSQA, SWA, TRT, RGN, and SUM.

The contents of areas in a Transaction dump depend on the dump type:
- Asynchronous Transaction Dump: The current task control block (TCB) and request block (RB) in the dump are for the dump task, rather than for the failing task. For additional address spaces in the dump, the TCB and RB are for the dump task.
- Synchronous Transaction Dump: The current TCB and RB in the dump are for the failing task.

Reference

See *z/OS MVS IPCS Commands* for examples of IPCS output formatted from Transaction Dumps.

Customizing Transaction Dump Contents

You can customize the contents of a Transaction dump to meet the needs of your installation. For example, you might want to add areas to be dumped, reduce the dump size, or dump Hiperspaces. In most cases, you will customize the contents of a Transaction dump or summary dump through the SDATA parameter of the IEATDUMP macro.

Hiperspaces

Transaction Dumps do not include Hiperspaces. To include Hiperspace data in a Transaction Dump, you have to write a program to copy data from the Hiperspace into address space storage that is being dumped.

Adding Areas

If the dump, as requested, will not contain all the needed areas, see one of the following for ways to add the areas:
- “Customized Contents Using the SDATA Parameter”
- “Contents of Summary Dumps in Transaction Dumps” on page 3-11

Customized Contents Using the SDATA Parameter

The IBM-supplied default contents and the contents available through customization are detailed in Table 3-1. The tables show dump contents alphabetically by the parameters that specify the areas in the dumps. Before requesting a dump, decide what areas will be used to diagnose potential errors. Find the areas in the tables. The symbols in columns under the dump indicate how the area can be obtained in that dump. The symbols are:

D IBM-supplied default contents

M Available on the macro that requests the dump

P Available in the parmlib member that controls the dump options

X Available on the CHNGDUMP operator command that changes the options for the dump type

blank No symbol indicates that the area cannot be obtained.

Note: System operator commands and assembler macros use the parameters in the table to specify dump contents.

The order of the symbols in the following table is not important.

Table 3-1. Customizing Transaction Dump Contents through the SDATA Parameter

SDATA Parameter Option	Dump Contents	Transaction dump for IEATDUMP Macro
ALLNUC	The DAT-on and DAT-off nucleuses	M P X
CSA	Common service area (CSA) (that is, subpools 227, 228, 231, 241)	M P X
DEFS	Default areas LSQA, NUC, PSA, RGN, SQA, SUM, SWA, TRT	M
ALL	CSA, GRSQ, LPA, NUC, RGN, SQA, SUM, SWA, TRT	X

Table 3-1. Customizing Transaction Dump Contents through the SDATA Parameter (continued)

SDATA Parameter Option	Dump Contents	Transaction dump for IEATDUMP Macro
GRSQ	Global resource serialization control blocks for the task being dumped: • Global queue control blocks • Local queue control blocks	M P X
IO	Input/output supervisor (IOS) control blocks for the task being dumped: • EXCPD • UCB	D
LPA	Active link pack area (LPA): module names and contents	M P X
LSQA	Local system queue area (LSQA) allocated for the address space (that is, subpools 203 - 205, 213 - 215, 223 - 225, 229, 230, 233 - 235, 249, 253 - 255)	D M P X
NUC	Read/write portion of the control program nucleus (that is, only the non-page-protected areas of the DAT-on nucleus), including: • CVT • LSQA • PSA • SQA	M P X
PSA	Prefixed save areas (PSA) for the processor at the time of the error or the processor at the time of the dump	D M P
RGN	Allocated pages in the private area of each address space being dumped, including subpools 0 - 127, 129 - 132, 203 - 205, 213 - 215, 223 - 225, 229, 230, 236, 237, 244, 249, 251 - 255. Also, allocated eligible storage above the 2–gigabyte address.	M P X

Table 3-1. Customizing Transaction Dump Contents through the SDATA Parameter (continued)

SDATA Parameter Option	Dump Contents	Transaction dump for IEATDUMP Macro
SQA	System queue area (SQA) allocated (that is, subpools 226, 239, 245, 247, 248)	D M P X
SUM	Summary dump (See “Contents of Summary Dumps in Transaction Dumps” on page 3-11.)	D M P X
SWA	Scheduler work area (SWA) (that is, subpools 236 and 237)	M P X
TRT	System trace, generalized trace facility (GTF) trace, and master trace, as available	D M P X
Default system data	Instruction address trace, if available	D
Default system data	Nucleus map and system control blocks, including: • ASCB for each address space being dumped • ASVT • Authorization table for each address space • CVT, CVT prefix, and secondary CVT (SCVT) • Entry tables for each address space • GDA • JSAB of each address space being dumped • Linkage stack • Linkage table for each address space • PCCA and the PCCA vector table • TOT • TRVT • UCB	D
Default system data	DFP problem data, if DFP Release 3.1.0 or a later release is installed	D
Default system data	Storage for the task being dumped and program data for all of its subtasks	D
Default system data	Storage: 4 kilobytes before and 4 kilobytes after the address in the PSW at the time of the error	D

Contents of Summary Dumps in Transaction Dumps

Request a summary dump for two reasons:

1. The SUM parameter requests many useful, predefined areas with one parameter.
2. The system does not write dumps immediately for requests from ASYNC requests. Therefore, system activity destroys much needed diagnostic data. When SUM is specified, the system saves copies of selected data areas at the time of the request, then includes the areas in the Transaction dump when it is written.

Summary dump does not contain volatile system information. The system writes this summary dump before returning control to the dump requester; the summary information is saved for each address space being dumped.

Customizing Transaction Dump Contents through Summary Dumps: The Summary Dump contents are as follows:

1. **Address space identifier (ASID) record** for the address space of the dump task
2. **Control blocks** for the recovery termination manager (RTM):
   - **RTM2WA** associated with all TCBs in the dumped address space
3. **Dump header**, mapped by AMDDATA

   See *z/OS MVS Data Areas, Vol 1 (ABEP-DALT)* for the AMDDATA mapping.
4. 4 kilobytes before and 4 kilobytes after:
   - All valid unique addresses in the PSWs in the RTM2WAs shown in the dump
   - All valid unique addresses in the registers in the RTM2WAs shown in the dump
5. **Supervisor control blocks**:
   - Current® linkage stack
   - Primary address space number (PASN) access list
   - Work unit access list

References

See the following for information about control blocks listed in the above table:
- *z/OS MVS Data Areas, Vol 1 (ABEP-DALT)*
- *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)*
- *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*
- *z/OS MVS Data Areas, Vol 4 (RD-SRRA)*
- *z/OS MVS Data Areas, Vol 5 (SSAG-XTLST)*

Customizing Contents Through Operator Commands

The dump options list for Transaction Dumps can be customized through a CHNGDUMP operator command by all the ways shown in Table 3-2 on page 3-12.

Nucleus Areas in Dumps

Dump options control the parts of the nucleus that appear in a dump. A diagnostician seldom needs to analyze all the nucleus. An installation can eliminate nucleus areas from dumps. If the IBM-supplied defaults are used, Transaction dump for an IEATDUMP macro contains the nucleus map and certain control blocks.

If no nucleus changes have been made, an installation should obtain one copy of the DAT-off nucleus for use with all dumps (SVC Dumps, SYSMDUMPs, and Transaction dumps). To obtain this nucleus, enter a DUMP operator command with

SDATA=ALLNUC and no other SDATA options. The nucleus does not change from one IPL to another, so one dump can be used again and again.

DAT, dynamic address translation, is the hardware feature that enables virtual storage. In the DAT-on part of the nucleus, the addresses are in virtual storage; in the DAT-off part of the nucleus, the addresses are in central storage.

Table 3-2. Customizing Transaction Dump Contents through Operator Commands

Customization	Effect	Example
Updating IEADMR00 parmlib member	**Change occurs**: At system initialization **What changes**: This command establishes the dump options for Transaction dumps for IEATDUMP macros and SDATA. See "Contents of Transaction Dumps" on page 3-7 for the list.	To add the link pack area (LPA) to all Transaction dumps for IEATDUMP macros and SDATA, while keeping the local system queue area (LSQA) and trace data, change the line in IEADMR00: `SDATA=(LSQA,TRT,LPA)`
Adding the CHNGDUMP operator command in IEACMD00 parmlib member	**Change occurs**: At system initialization **What changes**: This command establishes the dump options for Transaction dumps for IEATDUMP macros and SYSMDUMPS. See "Contents of Transaction Dumps" on page 3-7 for the list.	To add the link pack area (LPA) to all Transaction dumps for IEATDUMP macros and SYSMDUMP, while keeping the local system queue area (LSQA) and trace data, add the following command to IEACMD00: `CHNGDUMP SET,SYSMDUMP=(LPA)`

Table 3-2. Customizing Transaction Dump Contents through Operator Commands (continued)

Customization	Effect	Example

Entering CHNGDUMP operator command with SYSMDUMP parameter on a console with master authority	**Change occurs**: Immediately when command is processed
	
	What changes:
	
	For the ADD mode: CHNGDUMP options are added to the current Transaction dump options list and to any options specified in the macro or operator command that requested the dump. The options are added to all Transaction dumps for IEATDUMP macros and SYSMDUMP, until another CHNGDUMP SYSMDUMP operator command is entered.
	
	For the OVER mode: CHNGDUMP options are added to the current Transaction dump options list. The system ignores any options specified in the macro or operator command that requested the dump. The options override all Transaction dumps for the IEATDUMP macro and SYSMDUMP, until a CHNGDUMP SYSMDUMP,ADD operator command is entered.
	
	For the DEL option: CHNGDUMP options are deleted from the Transaction dump options list.
	
	When more than one CHNGDUMP operator command with IEATDUMP is entered, the effect is cumulative.
		To add the LPA to all Transaction dumps for the IEATDUMP macro and SYSMDUMP, until changed by another CHNGDUMP SYSMDUMP, enter:
		`CHNGDUMP SET,SYSMDUMP=(LPA)`
		
		To add the CHNGDUMP IEATDUMP options list to all Transaction dumps:
		`CHNGDUMP SET,SYSMDUMP,ADD`
		
		To override all Transaction dumps with the CHNGDUMP IEATDUMP options list:
		`CHNGDUMP SET,SYSMDUMP,OVER`
		
		To remove LPA from the IEATDUMP options list:
		`CHNGDUMP DEL,SYSMDUMP=(LPA)`

Analyzing Summary Transaction Dumps

The SUMDUMP or SUM option on the IEATDUMP macro causes Transaction dump to produce a summary dump, which can be formatted using the IPCS SUMDUMP subcommand. IBM strongly recommends that you view the SUMDUMP output prior to investigating the usual portions of the dump. Each summary dump record is indicated by the header “----tttt---- RECORD ID X'nnnn',” where *tttt* is the title for the type of SUMDUMP output, and *nnnn* is the hexadecimal record identifier assigned to the type.

The summary dump is formatted by the IPCS VERBEXIT SUMDUMP subcommand and has an index which describes what the summary contains.

Note: During Transaction dump processing, the system sets some tasks in the requested address space non-dispatchable; non-dispatchable tasks in the dump may have been dispatchable at the time of the problem.

Reference

See the SMDLR control block in *z/OS MVS Data Areas, Vol 4 (RD-SRRA)* for the record id values.

SUMDUMP Output for IEATDUMP

The SUMDUMP output can contain information that is not available in the remainder of the Transaction dump if options such as region, LSQA, nucleus, and LPA were not specified in the dump parameters.

For each address space dumped, the SUMDUMP output is preceded by a header with the ASID, plus the jobname and stepname for the last task created in the address space. The SUMDUMP output contains RTM2 work areas for tasks in address spaces that are dumped. Many of the fields in the RTM2WA provide valuable debugging information.

The summary dump data is dumped in the following sequence:

1. The ASID record is dumped for the address space.
2. All RTM2 work areas pointed to by all TCBs in this address space are dumped.
3. An address range table is built containing the following ranges, pointed to by the RTM2WA:
   - 4K before and after the PSW at the time of error (RTM2NXT1)
   - 4K before and after each register at the time of error (RTM2EREG).

   Duplicate storage is eliminated from this address range table to reduce the amount of storage dumped. In the summary dump output, this storage is written in ascending order to make finding addresses easier. This storage is dumped with record ID X'30'.

Analyzing a Transaction Dump

This section shows you how to use IPCS to analyze a Transaction Dump. You would analyze a Transaction dump because of one of the following:

- Dump output from the IPCS STATUS FAILDATA subcommand did not contain data for the abend being diagnosed.
- The problem involved multiple abends.
- The dump was taken but does not contain abend-related information.

This section contains the following topics, which, if followed in order, represent the procedure for analyzing a Transaction Dump:

- “Formatting the Transaction Dump Header” on page 3-15
- “Looking at the Dump Title” on page 3-15
- “Displaying the Time and Type of Dump” on page 3-17
- “Locating Error Information” on page 3-17
- “Analyze TCB Structure” on page 3-20
- “Examining the LOGREC Buffer” on page 3-22
- “Examining the System Trace” on page 3-25
- “Looking at the Registers” on page 3-25
- “Other Useful Reports for Transaction Dump Analysis” on page 3-26

Specifying the Source of the Dump

The first step in analyzing the dump is to specify the source of the dump that IPCS should format. In the IPCS dialog choose option 0 (DEFAULTS) and specify the name of the Transaction dump data set on the “Source” line.

```
------------------------ IPCS Default Values --------------------------------
COMMAND ===> _
  You may change any of the defaults listed below.
  If you change the Source default, IPCS will display the current default
  Address Space for the new source and will ignore any data entered in
  the Address Space field.

  Source  ==> DSN('D46IPCS.TDUMP.CSVLLA.DUMP002')
  Address Space   ==>                        Ignored if Source is changed.
  Message Routing ==> NOPRINT TERMINAL
  Message Control ==> FLAG(WARNING)
  Display Content ==> NOMACHINE REMARK REQUEST NOSTORAGE SYMBOL

Press ENTER to update defaults.
Use the END command to exit without an update.
```

Press Enter to register the new default source name. Then press PF3 to exit the panel.

You can also use the SETDEF subcommand to specify the source. For the dump in the preceding example, enter:

```
SETDEF DSNAME('D46IPCS.TDUMP.CSVLLA.DUMP002')
```

IPCS does not initialize the dump until you enter the first subcommand or IPCS dialog option that performs formatting or analysis. At that time IPCS issues message BLS18160D to ask you if summary dump data can be used by IPCS. The summary dump data should always be used for a Transaction dump because it is the data captured closest to the time of the failure. If you do not allow IPCS to use summary dump data, other data captured later for the same locations will be displayed, if available. Such data is less likely to be representative of the actual data at these storage locations at the time of the failure.

Formatting the Transaction Dump Header

The Transaction dump header contains the following information:
- SDWA
- Dump title, error identifier, and time of the dump
- Requestor of dump

Format data in the header of a Transaction dump using the following IPCS subcommands:

LIST TITLE
STATUS FAILDATA
STATUS REGISTERS
STATUS WORKSHEET

The following sections give examples of how to use these IPCS subcommands (or IPCS dialog options, where applicable) to obtain the desired information.

Looking at the Dump Title

The dump title tells you the component name, component identifier and module name. You can find the dump title using the following IPCS subcommands:

LIST TITLE
STATUS WORKSHEET

You can also obtain the STATUS WORKSHEET report through option 2.3 of the IPCS dialog. First choose option 2 (ANALYSIS) from the primary option menu:

```
--------------------- z/OS 01.02.00 IPCS PRIMARY OPTION MENU ----------------
OPTION ===> 2
                                                     ********************
 0DEFAULTS  - Specify default dump and options      * USERID   - IPCSU1
 1BROWSE    - Browse dump data set                  * DATE     - 97/06/08
 2ANALYSIS  - Analyze dump contents                 * JULIAN   - 97.160
 3SUBMIT    - Submit problem analysis job to batch  * TIME     - 16:43
 4COMMAND   - Enter subcommand, CLIST or REXX exec  * PREFIX   - IPCSU1
 5 UTILITY   - Perform utility functions             * TERMINAL - 3278
 6DUMPS     - Manage dump inventory                 * PF KEYS  - 24
 TTUTORIAL  - Learn how to use the IPCS dialog      ********************
 XEXIT      - Terminate using log and list defaults

Enter END command to end the IPCS dialog.
```

Then choose option 3 (WORKSHEET) from the analysis of dump contents menu:

```
-------------------- IPCS MVS ANALYSIS OF DUMP CONTENTS ---------------------
OPTION  ===> 3
To display information, specify the corresponding option number.

 1SYMPTOMS    - Symptoms                                *****************
 2STATUS      - System environment summary              * USERID  - IPCSU1
 3WORKSHEET   - System environment worksheet            * DATE    - 97/06/08
 4SUMMARY     - Address spaces and tasks                * JULIAN  - 97.160
 5CONTENTION  - Resource contention                     * TIME    - 16:44
 6COMPONENT   - MVS component data                      * PREFIX  - IPCSU1
 7TRACE       - Trace formatting                        * TERMINAL- 3278
 8STRDATA     - Coupling Facility structure data        * PF KEYS - 24
                                                          *****************
 Enter END command to terminate MVS dump analysis.
```

IPCS displays a new panel with information similar to that in Figure 3-2 on page 3-17. The dump title is labelled at the top of the STATUS WORKSHEET report. The dump title is “Compon=Program Manager Library-Lookaside, Compid=SC1CJ, Issuer=CSVLLBLD.”

Reference

See *z/OS MVS Diagnosis: Reference* for an explanation of dump titles.

```
IPCS OUTPUT STREAM ------------------------------------------ LINE 0 COL
COMMAND ===>                                                  SCROLL ===
  ****************************** TOP OF DATA ****************************


                             MVS Diagnostic Worksheet

Dump Title: COMPON=PROGRAM MANAGER LIBRARY-LOOKASIDE,COMPID=SC1CJ,
             ISSUER=CSVLLBLD

CPU Model 3090 Version FF Serial no. 170067 Address 01
Date: 05/05/1997   Time: 14:15:31  Local

Original dump dataset: APL.TDUMP.Dxxxxx.Txxxxxx

Information at time of entry to SVCDUMP:

HASID 001B  PASID 001B  SASID 001B  PSW 070C1000 8001AE5A

CML ASCB address 00000000  Trace Table Control Header address 7FFE3000
Dump ID: 001 Error ID: Seq 00051  CPU 0000  ASID X'001B'  Time 14:15:30.6
```

Figure 3-2. STATUS WORKSHEET Subcommand Sample Output — Dump Title

STATUS WORKSHEET also displays the error ID. In Figure 3-2, the dump ID is 001, error ID is sequence number 00051, ASID=X'001B', and processor 0000. Use this dump ID to match messages in SYSLOG and LOGREC records to the dump.

Displaying the Time and Type of Dump

The IPCS subcommand STATUS SYSTEM identifies

- The time of the dump
- The program requesting the dump

The IPCS dialog does not have a menu option for STATUS SYSTEM. Instead you must enter the subcommand.

Figure 3-3 is an example of a STATUS SYSTEM report.

```
SYSTEM STATUS:
  Nucleus member name: IEANUC01
   I/O configuration data:  NOT AVAILABLE
  Sysplex name: SYSPL1
  TIME OF DAY CLOCK: A7AD70E0 F144B700  06/23/1997 08:43:59.533131 local
  TIME OF DAY CLOCK: A7ADA685 F144B700  06/23/1997 12:43:59.627339 GMT
  Program Producing Dump: SYSMDUMP
  Program Requesting Dump: IEAVTDMP
```

Figure 3-3. Sample Output from the STATUS SYSTEM Subcommand

Locating Error Information

Use the IPCS subcommand STATUS FAILDATA to locate the specific instruction that failed and to format all the data in a Transaction dump related to the software failure. This report gives information about the CSECT involved in the failure, the component identifier, and the PSW address at the time of the error.

Note: For non-ABEND Transaction dumps, use SUMMARY FORMAT or VERBEXIT LOGDATA instead of STATUS FAILDATA.

Transaction Dump

Choose option 4 (COMMAND) from the IPCS primary option menu and enter the following command:

```
------------------------- IPCS Subcommand Entry -------------------------------
Enter a free-form IPCS subcommand, CLIST, or REXX exec invocation below:

===> STATUS FAILDATA
```

Use the PF keys to scroll up and down through the report. The following sections describe parts of the report.

Identifying the Abend and Reason Codes

Under the heading “SEARCH ARGUMENT ABSTRACT”, you will find the abend code and, if provided, an abend reason code.

```
:
 SEARCH ARGUMENT ABSTRACT

  PIDS/5752SC1CJ RIDS/CSVLLCRE#L RIDS/CSVLLBLD AB/S0FF0 REGS/09560 REGS/ 06026
   RIDS/CSVLLBLD#R

   Symptom             Description
   -------             -----------
   PIDS/5752SC1CJ      Program id: 5752SC1CJ
   RIDS/CSVLLCRE#L     Load module name: CSVLLCRE
   RIDS/CSVLLBLD       Csect name: CSVLLBLD
   AB/S0FF0            System abend code: 0FF0
   REGS/09560          Register/PSW difference for R09: 560
   REGS/06026          Register/PSW difference for R06: 026
   RIDS/CSVLLBLD#R     Recovery routine csect name: CSVLLBLD
:
```

Figure 3-4. Search Argument Abstract in the STATUS FAILDATA Report

In Figure 3-4, the abend code is X'FF0' with no reason code. See *z/OS MVS System Codes* for a description of the abend code and reason code.

The following IPCS reports also provide the abend and reason codes:
- VERBEXIT LOGDATA
- STATUS WORKSHEET
- VERBEXIT SYMPTOMS

Finding the System Mode

Below the “SEARCH ARGUMENT ABSTRACT“ section is information describing the system mode at the time of the error.

⋮

```
Home ASID: 001B    Primary ASID: 001B    Secondary ASID: 001B
PKM: 8000          AX: 0001              EAX: 0000

RTM was entered because a task requested ABEND via SVC 13.
The error occurred while: an SRB was in control.
No locks were held.
No super bits were set.
```

⋮

Figure 3-5. System Mode Information in the STATUS FAILDATA Report

The line that starts with “The error occurred...” tells you if the failure occurred in an SRB or TCB. In the example in Figure 3-5, the error occurred while an SRB was in control, which means you need to look under the heading SEARCH ARGUMENT ABSTRACT (see Figure 3-4 on page 3-18) to find the CSECT and load module names. This is the module in which the abend occurred.

If an SRB service routine was in control, look under the heading SEARCH ARGUMENT ABSTRACT for the CSECT and load module names. This is the failing module. that is nonzero.

If the error had occurred while a TCB was in control, you would find the failing TCB by formatting the dump using the IPCS subcommand SUMMARY TCBERROR. See “Analyze TCB Structure” on page 3-20.

Identifying the Failing Instruction

The STATUS FAILDATA report also helps you find the exact instruction that failed. This report provides the PSW address at the time of the error and the failing instruction text. Note that the text on this screen is not always the failing instruction text. Sometimes the PSW points to the place where the dump was taken and not the place where the error occurred.

In Figure 3-6 on page 3-20, the PSW at the time of the error is X'11E6A3C' and the instruction length is 4-bytes; therefore, the failing instruction address is X'11E6A38'. The failing instruction is 927670FB.

```
:
 OTHER SERVICEABILITY INFORMATION

   Recovery Routine Label:  CSVLEBLD
   Date Assembled:          88245
   Module Level:            JBB3313
   Subfunction:             LIBRARY-LOOKASIDE

 Time of Error Information

   PSW: 070C0000 811E6A3C   Instruction length: 04   Interrupt code: 0004
   Failing instruction text: E0009276 70FB5030 70F8D7F7

   Registers 0-7
   GR: 0002A017 00FBE800 00000000 00000076  00000C60 00FBE600 0002A016
00000017
   AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
   Registers 8-15
   GR: 012221A8 811E69F8 00000001 30000000  00FD82C8 811CAD90 011E69D0 011E69D0
   AR: 00000000 00000000 00000000 00000000  00000000 00000000 00005F60 00000000
:
```

Figure 3-6. Time of Error Information in the STATUS FAILDATA Report

The failing instruction text displayed in this report is always 12 bytes, 6 bytes before and 6 bytes after the PSW address. In this example, the failing instruction, 927670FB, is an MVI of X'76' to the location specified by register 7 + X'FB'.

Register 7 at the time of the error, shown under **Registers 0-7** above, contained a X'00000017'. The attempted move was to storage location X'112'. The first 512 bytes of storage are hardware protected. Any software program that tries to store into that area without authorization will receive a protection exception error and a storage protection exception error.

Reference

See *Principles of Operation* for information about machine language operation codes, operands, and interruption codes.

To find the module that abnormally terminated and the offset to the failing instruction, use the WHERE command. WHERE can identify the module or CSECT that the failing PSW points to.

Analyze TCB Structure

If a TCB was in control at the time of the error, use the IPCS subcommand SUMMARY TCBERROR to look at the TCB information and find the failing component. SUMMARY TCBERROR summarizes the control blocks for the failing address space. (To see all the fields in the control blocks, use SUMMARY FORMAT.) Scan the completion codes (field CMP) for each TCB to find the correct TCB. This report displays RBs from newest to oldest.

Figure 3-7 on page 3-21 is an example of SUMMARY TCBERROR output. In this example the TCB at address 008E9A18 has a completion code of X'0C1.' The error occurred under this TCB. Once you have identified the failing TCB, you can follow the RB chain to the failing program.

```
⋮
     NAME..... IEFSD060  ENTPT.... 00DA6308

  PRB: 008FFED0
     WLIC..... 00020006  FLCE.... 00C534A0  OPSW..... 070C2000 00DC766A
     LINK..... 008FFA10

  CDE: 00C534A0
     NAME..... IEESB605  ENTPT.... 00DC7000

  TCB:  008FF0D0
     CMP...... 00000000  PKF...... 80        LMP...... FF        DSP...... FF
     TSFLG.... 00        STAB..... 008FD210  NDSP..... 00002000
     JSCB..... 008FF4FC  BITS..... 00000000  DAR...... 00
     RTWA..... 00000000  FBYT1.... 00
     Task non-dispatchability flags from TCBFLGS4:
      Top RB is in a wait
     Task non-dispatchability flags from TCBFLGS5:
      Secondary non-dispatchability indicator
     Task non-dispatchability flags from TCBNDSP2:
      Transaction Dump is executing for another task
  PRB: 008E9F20
     WLIC..... 00020001  FLCDE.... 00C4CA38  OPSW..... 070C1000  00DAC66E
     LINK..... 018FF0D0

  CDE: 00C4CA38
     NAME..... IEFIIC    ENTPT.... 00DA6000
  TCB: 008E9A18
     CMP...... 940C1000  PKF...... 80        LMP...... FF        DSP...... FF
     TSFLG.... 20        STAB..... 008FD180  NDSP..... 00000000
     JSCB..... 008FF33C  BITS..... 00000000  DAR...... 01
     RTWA..... 7FFE3090  FBYT1.... 08
```

Figure 3-7. An Example of the SUMMARY TCBERROR Report (Part 1 of 2)

```
SVRB: 008FD7A8
     WLIC..... 00020000  FLCDE.... 00000000  OPSW..... 070C1000  82569B38
     LINK..... 008FD638

  SVRB: 008FD7A8
     WLIC..... 00020000  FLCDE.... 00000000  OPSW..... 070C1000  82569B38
     LINK..... 008FD638

  PRB: 008E9750
     WLIC..... 00020033  FLCDE.... 14000000  OPSW..... 070C1000  80CE9AEE
     LINK..... 008FD638

  SVRB: 008FD638
     WLIC..... 0002000C  FLCDE.... 00000000  OPSW..... 070C1000  825E9768
     LINK..... 008FD4C8

  SVRB: 008FD4C8
     WLIC..... 00020001  FLCDE.... 00000000  OPSW..... 070C0000  00C47D52
     LINK..... 008FD358

  SVRB: 008FD358
     WLIC..... 00020053  FLCDE.... 00000000  OPSW..... 075C0000  00D64EOC
     LINK..... 008FF4D8

  PRB: 008FF4D8
     WLIC..... 00020014  FLCDE.... 008FF3D8  OPSW..... 078D0000  00006EF2
     LINK..... 008E9A18

  CDE: 008FF3D8
     NAME..... SMFWT     ENTPT.... 00006EB0
```

Figure 3-7. An Example of the SUMMARY TCBERROR Report (Part 2 of 2)

In this example, the most current RB is the SVRB at address 008FD7A8. This is the Transaction dump's RB. The ESTAE's RB is the PRB at 008E9750. The ESTAE issued an SVC 33. The RB for the recovery termination manager (RTM) is the SVRB at 008FD638. RTM issued an SVC C to attach the ESTAE. The X'0C1' abend occurred under the SVRB at 008FD4C8. The last interrupt was a 1 at the address indicated in the old PSW field (OPSW). The next RB in the chain shows an SVC X'53' (SMFWTM) had been issued. This is the code the X'0C1' occurred in.

For an asynchronous dump, the abnormally terminating TCB can generally be found by scanning for a nonzero completion code. If there is no code, scan the system trace for the abend. The trace identifies the ASID number and TCB address for each entry. See “Examining the System Trace” on page 3-25.

Use the STATUS or the STATUS REGS subcommand to find the data set name and the module name of the Transaction dump requester.

Examining the LOGREC Buffer

Use the IPCS subcommand VERBEXIT LOGDATA to view the LOGREC buffer in a dump. This report might repeat much of the information contained in the STATUS FAILDATA report, but it helps to identify occasions when multiple error events caused the software failure.

The example in Figure 3-8 on page 3-23 shows how multiple errors can appear in the LOGREC buffer. Abend X'0D5' is the first abend and X'058' is the second. Always check for multiple errors in the VERBEXIT LOGDATA report that are in the same address space or a related address space and are coincident with or precede the Transaction Dump.

```
 TYPE:  SOFTWARE RECORD      REPORT:  SOFTWARE EDIT REPORT          DAY.YEAR
        (PROGRAM INTERRUPT)                           REPORT DATE: 235.97
 SCP:   VS 2 REL 3                                     ERROR DATE: 126.97
                             MODEL:   3090                          HH:MM:SS.TH
                             SERIAL:  272804                  TIME: 13:27:59.86

 JOBNAME: LSCMSTR

ERRORID: SEQ=01196  CPU=0042  ASID=000C  TIME=13:27:59.6

 SEARCH ARGUMENT ABSTRACT

   PIDS/####SC1C5 RIDS/NUCLEUS#L RIDS/IEAVEDS0 AB/S00D5 PRCS/00000021 REGS/0F120
   RIDS/IEAVEDSR#R

   SYMPTOM             DESCRIPTION
   -------             -----------
   PIDS/####SC1C5      PROGRAM ID: ####SC1C5
   RIDS/NUCLEUS#L      LOAD MODULE NAME: NUCLEUS
   RIDS/IEAVEDS0       CSECT NAME: IEAVEDS0
   AB/S00D5            SYSTEM ABEND CODE: 00D5
   PRCS/00000021       ABEND REASON CODE: 00000021
   REGS/0F120          REGISTER/PSW DIFFERENCE FOR R0F: 120
   RIDS/IEAVEDSR#R     RECOVERY ROUTINE CSECT NAME: IEAVEDSR

 OTHER SERVICEABILITY INFORMATION

   RECOVERY ROUTINE LABEL:  IEAVEDSR
   DATE ASSEMBLED:          08/23/89
   MODULE LEVEL:             UY41669
   SUBFUNCTION:             DISPATCHER

 TIME OF ERROR INFORMATION

   PSW: 440C0000 80FEFC56   INSTRUCTION LENGTH: 04   INTERRUPT CODE: 0021
   FAILING INSTRUCTION TEXT: 1008B777 1008B225 000007FE
   TRANSLATION EXCEPTION IDENTIFICATION: 00000041
   REGISTERS 0-7
   GR: 00000041 00F9A0C0 00000000 00000000  00000000 008DE188 008E8C78 00000001
   REGISTERS 8-15
   GR: 00F97280 0103AB6A 00FF1B08 008DE188  0000000C 000C0041 80FF6510 00FEFB36

   HOME ASID: 000C    PRIMARY ASID: 000C    SECONDARY ASID: 000C
   PKM: 8000          AX: 0001

   RTM WAS ENTERED BECAUSE OF A PROGRAM CHECK INTERRUPT.
   THE ERROR OCCURRED WHILE A LOCKED OR DISABLED ROUTINE WAS IN CONTROL.
   NO LOCKS WERE HELD.
   SUPER BITS SET: PSADISP  - DISPATCHER
```

Figure 3-8. Sample Output from the VERBEXIT LOGDATA Subcommand (Part 1 of 2)

```
.
.
.
TYPE:  SOFTWARE RECORD      REPORT:  SOFTWARE EDIT REPORT         DAY.YEAR
       (SVC 13)                                      REPORT DAT 235.97
SCP:   VS 2 REL 3                                     ERROR DAT 126.97
                            MODEL:   3090                       HH:MM:SS.TH
                            SERIAL:  272804                 TIM 13:27:59.97

JOBNAME: LSCMSTR
ERRORID: SEQ=01197  CPU=0000  ASID=000C  TIME=13:27:59.6

SEARCH ARGUMENT ABSTRACT

  AB/S0058

  SYMPTOM             DESCRIPTION
  -------             -----------
  AB/S0058            SYSTEM ABEND CODE: 0058


SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE

  PROGRAM ID
  LOAD MODULE NAME
  CSECT NAME
  RECOVERY ROUTINE CSECT NAME
  RECOVERY ROUTINE LABEL
  DATE ASSEMBLED
  MODULE LEVEL
  SUBFUNCTION

TIME OF ERROR INFORMATION

  PSW: 470C8000 00FDC266   INSTRUCTION LENGTH: 02   INTERRUPT CODE: 000D
  FAILING INSTRUCTION TEXT: 00000000 0A0D0A06 00000000

  REGISTERS 0-7
  GR: 00A5D7A8 80058000 00000041 022DC780  008EDF00 008FBC7C 00F86A00 026E01F0
  REGISTERS 8-15
  GR: 026E0160 00000000 8001AC96 0001BC96  00000000 000188F0 8001B194 00000020

  HOME ASID: 000C    PRIMARY ASID: 000C    SECONDARY ASID: 000C
  PKM: 8000          AX: 0001

  RTM WAS ENTERED BECAUSE AN SVC WAS ISSUED IN AN IMPROPER MODE.
  THE ERROR OCCURRED WHILE AN ENABLED RB WAS IN CONTROL.
  NO LOCKS WERE HELD.
  NO SUPER BITS WERE SET.
```

Figure 3-8. Sample Output from the VERBEXIT LOGDATA Subcommand (Part 2 of 2)

When viewing the VERBEXIT LOGDATA report, skip the hardware records to view the software records. Search for the first software record.

The field “ERRORID=” gives the error identifier for the software failure. The error identifier consists of the sequence number, ASID, and time of the abend. By matching this identifier with error identifiers from other reports, you can tell if this is the same abend you have been analyzing or if it is a different abend. See “Interpreting Software Records” on page 14-19 for more information.

Examining the System Trace

The system trace table describes the events in the system leading up to the error. The trace table is helpful when the PSW does not point to the failing instruction, and to indicate what sequence of events preceded the abend.

IPCS option 2.7.4 formats the system trace. The report is long. IBM recommends scrolling to the end of the report, then backing up to find the trace entry for the abend. Type an M on the command line and press F8 to scroll to the bottom of the report.

After you find the entry for the abend, start at the PSW where the dump was taken and track the events in the table to find where the failing instruction is in the code.

The system trace report marks important or significant entries with an asterisk. In Figure 3-9 "*SVC D" in the "IDENT CD/D" column identifies the PSW where the program took the dump. Prior to the SVC D are three PGM (program check) entries. PGM 001 has an asterisk next to it, indicating that the program check was unresolved. The next entry, RCVY PROG, identifies a recovery program that failed because it issued the SVC D a few entries later. See Chapter 8, "System Trace" on page 8-1 to recognize significant entries in the system trace table.

```
PR  ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3
                                                 UNIQUE-4 UNIQUE-5 UNIQUE-6

 :

 01 0094 00AF7D18   DSP       070C0000 81EA7000  00000000 00000000 0000800C
 01 0094 00AF7D18   SVC    78 070C0000 81EA7048  00000002 00000278 00000000
 01 0094 00AF7D18  tdmpR   78 070C0000 81EA7048  00000000 00000278 03300D88
 01 0094 00AF7D18   PGM   010 070C0000 81EA704A  00040010 03300D8C
 01 0094 00AF7D18   PGM   011 070C0000 81EA704A  00040011 03300D8C
 01 0094 00AF7D18   SVC    77 070C2000 81EA7088  81EA7000 00000000 00050000
 01 0094 00AF7D18  tdmpR   77 070C2000 81EA7088  00000000 00000000 40000000
 01 0094 00AF7D18  *PGM   001 070C0000 83300FAA  00020001 03300D8C
 01 0094 00AF7D18  *RCVY PROG                    940C1000 00000001 00000000
 02 0001 00000000   I/O   1A2 070E0000 00000000  0080000E 060246C0 0C000001
 02 0054 00AD7300   SRB       070C0000 810537E0  00000054 00F3C9F8 00F3CA40
 01 0054 00AF7D18   SSRV  12D          810B9CEE  00AF7D18 000C0000 00000000
 01 0094 00AF7D18   SSRV  12D          810B9D0E  00AF7D18 000B0000 00000000
 01 0094 00AF7D18   DSP       070C0000 810BF664  00000000 00000000 40000000
 01 0094 00AF7D18  *SVC     D 070C0000 810BF666  00000040 00000000 40000000
 01 0054 00000000  SSRV   10F          00000000  00F83E80 00AD7300 00AC5040
```

Figure 3-9. An Example of Output from the IPCS Subcommand SYSTRACE

Looking at the Registers

Use the IPCS subcommand STATUS REGISTERS to display the registers for the TCBs and RBs. SUMMARY REGS gives the same information in a different format.

This report identifies the PSW, ASID and register values just as the STATUS FAILDATA report, but STATUS REGISTERS also gives the control register values.

```
CPU STATUS:

 PSW=070C1000  80FE5CFC  (RUNNING IN PRIMARY, KEY 0,  AMODE 31, DAT ON)
     DISABLED FOR PER
   ASID(X'001B') 00FE5CFC. IEANUC01.IEAVESVC+05FC IN READ ONLY NUCLEUS
  ASCB27 at F3FA00, JOB(LLA), for the home ASID
  ASXB27 at 9FDF00 for the home ASID. No block is dispatched
  HOME ASID: 001B PRIMARY ASID: 001B SECONDARY ASID: 001B

  GPR VALUES
      0-3  80000000  80FF0000  009FF5A0  00FC4E88
      4-7  009F8E88  009FD358  80FE5CD6  00F3FA00
      8-11 00000000  80FE579C  009FD418  7FFFE2C0
     12-15 7FFE0000  00006730  00FE6200  80014910

  ACCESS REGISTER VALUES
      0-3  7FFEA5CC  00000000  00000000  00000000
      4-7  00000000  00000000  00000000  00000000
      8-11 00000000  00000000  00000000  00000000
     12-15 00000000  00000000  00005F60  8210532A

ALET TRANSLATION
AR 00    Not translatable
AR 14    Not translatable
AR 15    Not translatable

  CONTROL REGISTER VALUES
      0-3  5EB1EE40  00A2007F  007CCDC0  8000001B
      4-7  0001001B  00C506C0  FE000000  00A2007F
      8-11 00000000  00000000  00000000  00000000
     12-15 0082E07B  00A2007F  DF880C71  7FFE7008
```

Figure 3-10. Sample of the STATUS REGISTERS Report

The example output in Figure 3-10 shows the address in the PSW is X'0FE5CFC', the ASID is X'1B', and the failing instruction is located in offset X'5FC' in the CSECT IEAVESVC in the module IEANUC01 in the nucleus. You can now browse the dump at this location and look at the specific failing instruction. You could also use the information about the registers to find out more about the error if the address in the PSW does not point to the failing instruction.

Other Useful Reports for Transaction Dump Analysis

To collect further Transaction dump data, use any of the following commands.

IPCS Subcommand	Information in the Report
STATUS CPU REGISTERS DATA CONTENTION	Data about the abend, current ASID, and task.
SUMMARY FORMAT	All fields in the TCBs and the current ASID.
TCBEXIT IEAVTFMT 21C.%	The current FRR stack.
LPAMAP	The entry points in the active LPA and PLPA.
VERBEXIT NUCMAP	A map of the modules in the nucleus when the dump was taken.
VERBEXIT SUMDUMP	The data dumped by the SUMDUMP option on the IEATDUMP macro.
VERBEXIT MTRACE	The master trace table.

IPCS Subcommand	Information in the Report
VERBEXIT SYMPTOMS	The primary and secondary symptoms if available.

Note: Use the VERBEXIT SYMPTOMS subcommand last in your Transaction dump analysis. Other subcommands can add symptoms to the dump header record. This ensures VERBEXIT SYMPTOMS provides all symptoms available from the dump.

Chapter 4. Stand-Alone Dump

Like a trip to the dentist. . . you only go when you have to, and you know it's gonna hurt.

The stand-alone dump program produces a stand-alone dump of storage that is occupied by one of the following:

- A system that failed.
- A stand-alone dump program that failed.

  Either the stand-alone dump program dumped itself — a **self-dump** —, or the operator loaded another stand-alone dump program to dump the failed stand-alone dump program.

The stand-alone dump program and the stand-alone dump together form what is known as the stand-alone dump service aid. The term stand-alone means that the dump is performed separately from normal system operations and does not require the system to be in a condition for normal operation.

The stand-alone dump program produces a high-speed, unformatted dump of central storage and parts of paged-out virtual storage on a tape device or a direct access storage device (DASD). The stand-alone dump program, which you create, must reside on a storage device that can be used to IPL.

Produce a stand-alone dump when the failure symptom is a wait state with a wait state code, a wait state with no processing, an instruction loop, or slow processing.

You create the stand-alone dump program that will dump the storage. Use the AMDSADMP macro to produce the following:

- A stand-alone dump program that resides on DASD with output directed to a tape volume or to a DASD dump data set
- A stand-alone dump program that resides on tape, with output directed to a tape volume or to a DASD dump data set.

A stand-alone dump supplies information that is needed to determine why the system or the stand-alone dump program failed.

Create different versions of the stand-alone dump program to dump different types and amounts of storage. You can create different versions of the stand-alone dump program by coding several AMDSADMP macros and varying the values of keywords on the macros.

This chapter covers the following topics, which describe how to use stand-alone dump:

- "Planning for Stand-Alone Dump" on page 4-2
- "Creating the Stand-Alone Dump Program" on page 4-6
- "Running the Stand-Alone Dump Program" on page 4-36
- "Running the Stand-Alone Dump Program in a Sysplex" on page 4-41
- "Copying, Viewing, and Printing Stand-Alone Dump Output" on page 4-44
- "Message Output" on page 4-49
- "Analyzing Stand-Alone Dump Output" on page 4-50

Planning for Stand-Alone Dump

There are several decisions you need to plan for a stand-alone dump. You implement most of these decisions when you create the stand-alone dump program, either when you code the AMDSADMP macro or when you assemble the macro. Some typical questions follow.

Should I take a stand-alone dump to DASD or to tape?

When choosing an output device for stand-alone dump, consider the need for operator intervention, the amount of operator intervention involved and the amount of time the system will be unavailable.

You can reduce the level of operator intervention during stand-alone dump processing by dumping to DASD. With an automation package set up to IPL the stand-alone dump program from DASD, stand-alone dump can be run from a remote site. When you dump to tape, an operator is required to handle other aspects of dumping, such as mounting or changing tapes.

The system will be unavailable when a stand-alone dump is taken. The amount of time the system is unavailable depends upon the size of the dump.

See “Dumping to a DASD Data Set” on page 4-22 for more information.

If I do dump to DASD, how much space do I need?

A lot. IBM recommends that you specify enough space to dump all of central storage, all of expanded storage, and those address spaces that are swapped out. The size of your dump output depends on your storage configuration and how much of that storage you choose to dump using the options of stand-alone dump. To capture the storage you are dumping, you need to allocate at least as many cylinders as a typical dump to tape occupies when it has been copied to DASD for IPCS processing. The maximum size of a DASD dump data set is 65,536 tracks.

If more space is required, a multi-volume DASD dump data set (called a dump group within stand-alone dump), can be defined. This data set can span up to 16 volumes of the same device type, thus increasing the maximum amount to about 48 GB using 3390s.

When using a multi-volume DASD dump data set, the device number of the first volume is specified to use all volumes of the data set. All volumes will be written concurrently by stand-alone dump. If unable to access all volumes of the data set or invalid control information is read from the data set during initialization, the data set will be rejected.

If you do not allocate enough space in your dump data set, the stand-alone dump program prompts the operator to continue dumping to another DASD dump data set or tape volume. You can continue dumping to any stand-alone dump supported device, however, once a tape device is selected, it must be used to complete the dump even though multiple volumes may be required.

IBM recommends that you allocate multiple dump data sets so that a complete stand-alone dump can be successfully completed.

Can I dump to multiple dump data sets?

Stand-alone dump **does** allow you to dump to multiple dump data sets. By coding the DDSPROMPT=YES keyword on the AMDSADMP macro, you can generate a stand-alone dump program that allows run-time dump data set prompting.

When the Stand-alone dump program is initiated, message AMD001A is issued to prompt the operator for an output device. If a DASD device is specified and run-time dump data set prompting is active, message AMD002A is issued to prompt the operator for a dump data set name. Providing the dump data set is validly allocated and initialized on the output device, the stand-alone dump program will use the dump data set name specified. If message AMD099I is issued indicating that the dump data set is full, the operator can continue dumping to any stand-alone dump supported DASD dump data set or tape device by replying to message AMD001A (and possibly AMD002A) again. Once the dump completes, message AMD104I is issued to indicate the entire set of devices and/or dump data sets that were used during the taking of the dump.

By coding DDSPROMPT=NO on the AMDSADMP macro, the stand-alone dump program is generated without run-time dump data set prompting. In this case, replying to message AMD001A with a DASD device will cause the stand-alone dump program to assume that the output dump data set is named SYS1.SADMP.

Notes:

1. The user must still use the AMDSADDD REXX utility to allocate and initialize the stand-alone dump dump data sets.
2. Users must be aware that the stand-alone dump program must be able to locate the dump data set on the device that is specified. Therefore, it is imperative that the necessary data set management steps be taken so that the stand-alone dump dump data sets will not be placed into a migrated state or moved to a different volume. The dump data sets must also be exempt from any space management processing that will release unused space.
3. You can continue a dump to any stand-alone dump supported device, however, once a tape device is selected, it must be used to complete the dump even though multiple tape volumes may be required.

For more information on dump data set processing, see the description of the DDSPROMPT keyword in . For more information on how to use multiple dump data sets with IPCS, see “Copying from Multiple Dump Data Sets” on page 4-46.

What can I name my DASD dump data sets?

A stand-alone dump dump data set can be any valid MVS data set name, however, stand-alone dump has two requirements that are checked at both generation time and run-time:

- The data set name must be 44 characters or less
- The data set name must contain the text ’SADMP’ as either part of, or as an entire data set qualifier

In addition, since the generation process does not perform any allocation on the output device or dump data set name, it is imperative that the user insure that the data set name specified on the OUTPUT= keyword matches exactly the dump data set name allocated by the AMDSADDD utility. The following are some additional rules to follow when specifying a dump data set name:

- The data set name specified should be fully qualified (without quotes)
- The alphabetic characters in the dump data set name should be specified as capital letters

How much of the system should I dump?

It depends. The situation will dictate the amount of information you need to diagnose the failure. IBM recommends that you assemble two AMDSADMP macros, one using the MINASID(ALL) option and the other using the MINASID(PHYSIN) option. Based on the situation encountered, you can then choose which AMDSADMP macro to use to create the stand-alone dump program. If you choose to create only one version of the stand-alone dump program, you should specify MINASID(ALL).

Specify MINASID(ALL) when you need all of the address spaces, particularly for hangs, enabled waits, and performance problems. The MINASID(ALL) option provides a more complete image of the system at the time the dump is taken; however, this option increases the run time of the stand-alone dump program and the DASD space required for the dump data set, if you are dumping to DASD.

Specify MINASID(PHYSIN) for failures involving coded waits, loops, and spin loops. The MINASID(PHYSIN) option reduces the overall run time of the stand-alone dump program, but it provides a less complete picture of the system, increasing the risk of missing necessary diagnostic information. This option dumps only the physically swapped in address spaces, thereby excluding some virtual storage from the dump. If you chose MINASID(ALL), you can get almost the same results as MINASID(PHYSIN) by manually terminating the dumping process with an external interrupt when message AMD108I is issued.

You can also use the DUMP keyword to control the amount of storage you want dumped. See “Using the Dump Keyword to Request Additional Storage” on page 4-17 for more information.

Reference

See *z/OS MVS System Messages, Vol 1 (ABA-AOM)* for more information about AMD108I.

When should I specify the dump tailoring options?

The most flexible way to specify the dump options for a stand-alone dump is to specify on the DUMP keyword of the AMDSADMP macro those areas of storage you want dumped **and** allow the operator who requests the dump to specify additional options. To prompt the operator for additional dump tailoring options, specify PROMPT on the AMDSADMP macro.

In most cases, it is best to define any installation specific dump options on the AMDSADMP macro and **not** specify the PROMPT keyword to simplify the dumping process.

See “Using the Dump Keyword to Request Additional Storage” on page 4-17 for more information.

What type of security does the stand-alone dump program require?

Once the stand-alone dump program is properly created on a DASD residence volume, it resides in the SYS1.PAGEDUMP.Vvolser data set. To ensure that the stand-alone dump program is available and processes successfully, do not delete the data set or move it to another volume or pack. To protect the stand-alone dump program in SYS1.PAGEDUMP.Vvolser, use a password or a security product, such as RACF. If the data set is not protected, unauthorized users can read the dump data in SYS1.PAGEDUMP.Vvolser. Also consider protecting the stand-alone dump macros, modules, and the output dump data sets from unauthorized modification.

See *z/OS Security Server RACF Security Administrator’s Guide* for more information about protecting a data set.

Should I use IEBGENER or the COPYDUMP subcommand to copy a dump to a data set?

To use the IPCS COPYDUMP subcommand, the IPCS environment must be established. Many operators, however, take a stand-alone dump so that the system programmer can view the dump. The operator, however, does not require IPCS on the system because the operator will not be viewing the dump. Therefore, the operator should use the IEBGENER utility to copy the dump to a data set accessible by the system programmer's system.

What is dumped when I run the stand-alone dump program?

The default dump contains all areas of central storage and some areas of virtual storage that are not backed by central storage.The output of the stand-alone dump program includes:

- The prefixed save areas (PSA)
- The nucleus and extended nucleus
- The system queue area (SQA) and the extended SQA
- The common service area (CSA) and the extended CSA
- Subpools 203-205, 213-215, 229, 230, 236, 237, 247, 248, and 249 for the eligible address spaces based on the specified MINASID option (PHYSIN or ALL)
- The local system queue area (LSQA) and the extended LSQA for eligible address spaces based on the specified MINASID option (PHYSIN or ALL)
- The dump title provided by the operator; otherwise, the dump is untitled
- The processor STORE STATUS information for each processor
- Central storage from address 0 to the top of main storage (some blocks may be missing because of offline storage elements)
- Instruction trace data created by the instruction address trace
- Virtual storage areas selected by the DUMP keyword, or selected by the operator at run-time.
- A message log, normally consisting of all console messages issued by the dump program, including suppressed messages. (To format and print the stand-alone dump message log, use the VERBEXIT SADMPMSG subcommand or the SADMPMSG option of the IPCS dialog.)
- Eligible address spaces based on the specified MINASID option (PHYSIN or ALL)
- Dump records summarizing the zeroed pages in the dump
- The full GTF address space
- Subpool 127 in the GRS address space
- Dataspaces whose names begin with ISG for the GRS address space
- All of DUMPSRV's dataspaces
- The full XCF address space
- All of XCF's dataspaces
- XES-related dataspaces for address spaces with an XES connection

Note that this list does not imply an order of the stand-alone dump process. During stand-alone dump processing, several different messages are issued to indicate the progress of the dumping:

- For real dump processing, AMD005I is issued.
- For both real and virtual dump processing, AMD095I is issued every 30 seconds, followed by message AMD056I indicating that dumping of virtual storage has

completed and AMD104I to indicate what output devices and/or dump data sets were used by the stand-alone dump program.

Can I use my current version of the stand-alone dump program to dump a new version of MVS?

Although the current level of the stand-alone dump program might be able to dump a new level of MVS successfully, it is not guaranteed. The new version of MVS may have changed such that the stand-alone dump program would not be able to locate vital information it needs to operate.

When migrating to a new version of MVS, IBM recommends that you generate a new version of the stand-alone dump program built from the new MVS system data sets. See “Using Two-Stage Generation when Migrating” on page 4-33 for more information.

Creating the Stand-Alone Dump Program

The first step in creating a stand-alone dump program is selecting a tape or DASDas the stand-alone dump IPL volume (residence volume). After you select the residence volume, you can create the stand-alone dump program. To create the stand-alone dump program, you:

1. Code the AMDSADMP macro. See .
2. Assemble the macro, placing the stand-alone dump programonto the residence volume in ready-to-load form. IBM recommends that you use one-step generation when building or creating a stand-alone dump program for the currently executing version of MVS. Use the two-stage generation to create multiple stand-alone dump programs and to create a new version of the stand-alone dump program when migrating to a new version of MVS. See “Generating the Stand-Alone Dump Program” on page 4-28.

MNOTES from the AMDSADMP Macro

The output listing from the assembly may contain error messages, called MNOTES, that describe errors made while coding the AMDSADMP macro. To respond to one of these messages, check the specification of the macro and run the assembly step again.

The meaning of the severity code is as follows:

8 Assembly processing ends
4 Warning
0 Informational

AMDSADMP: COMPACT=*compact* IS NOT ALLOWED. IT MUST BE YES OR NO. COMPACT=YES HAS BEEN USED.

Explanation: The system could not recognize the value specified on the COMPACT keyword. The stand-alone dump program will use the IDRC feature for the output tape if IDRC is installed.

Severity Code: 0.

AMDSADMP: CONSOLE ADDRESS *conad*
IS INVALID. IT MUST BE A DEVICE NUMBER. 001F IS SUBSTITUTED.

Explanation: The console address operand is not a valid device number of 3 or 4 hexadecimal digits.

Severity Code: 0.

AMDSADMP: CONSOLE PARM NOT DETECTED. DEFAULT (001F, 3278) WILL BE USED.

Explanation: Either the console parameter was not specified or it was not specified correctly on the continuation statement. The parameter was probably not continued correctly on the next defined statement. Continue the interrupted parameter or field beginning in any column from 4 through 16.

(See *z/OS MVS JCL Reference*. Read the topic covering ‘Continuing Statements’.)

Severity Code: 0.

AMDSADMP: CONSOLE TYPE *contp* **IS INVALID. IT MUST BE A 4 DIGIT NUMBER. 3278 HAS BEEN USED.**

Explanation: An incorrect console type was specified. Only 3277, 3278, 3279, or 3290 are acceptable.

Severity Code: 0.

AMDSADMP: DEFAULT OUTPUT DEVICE T0282 WILL BE USED.

Explanation: A device number was incorrectly specified, or was not specified, on the OUTPUT= parameter.

Severity Code: 0.

AMDSADMP: IPL=*ipl* **IS INVALID. FIRST CHARACTER MUST BE D OR T, AND HAS BEEN REPLACED WITH A D.**

Explanation: The IPL operand is incorrect. It is not prefixed with a 'D' or a 'T'.

Severity Code: 4.

AMDSADMP: IPL=*ipl* **IS TOO LONG. THE UNIT NAME WILL BE TRUNCATED.**

Explanation: The unit name can be at most 8 characters long.

Severity Code: 4.

AMDSADMP: IPLUNIT WAS NOT SPECIFIED OR IPL= TYPE (D OR T) WAS SPECIFIED INCORRECTLY. UNIT WILL BE DEFAULTED TO SYSDA.

Explanation: The IPL parameter should be specified as IPL=duuu, where ‘d’ is D for direct access or T for tape, and ‘uuu’ is a valid unit type or device number for the SADMP IPL volume as described by the UNIT=uuu JCL parameter.

System Programmer Response: A device number consists of 3 or 4 hexadecimal digits.

Severity Code: 0.

AMDSADMP: MSG=*msg* **IS INVALID. IT MUST BE ALL, ACTION, OR ALLASIDS. MSG=ALL HAS BEEN USED.**

Explanation: The MSG operand is not ALL, ACTION, or ALLASIDS.

Severity Code: 0.

AMDSADMP: DDSPROMPT=*ddsprompt* IS NOT ALLOWED. IT MUST BE YES OR NO. DDSPROMPT=YES HAS BEEN USED.

Explanation: The DDSPROMPT operand is incorrect. It must be either ‘YES’ or ‘NO’. DDSPROMPT=YES is assumed.

System Action: The SADMP program will be generated with run-time dump data set prompting active.

Severity Code: 0.

AMDSADMP: OUTPUT=*output* IS INCORRECT. IT MUST BE EITHER {T|D}UNIT OR (DUNIT,DATA SET NAME).

Explanation: The OUTPUT operand is incorrect. It must be specified in one of the following formats:
- A ‘T’ or a ‘D’ followed by a device number
- A ‘D’ followed by a device number and a data set name pair specified within parentheses.

System Action: Generation continues, using the default for the OUTPUT operand, T0282, regardless of the format used.

System Programmer Response: The output device must be specified as a 3-digit or 4-digit device number. You can change the OUTPUT parameter at run time, if the default is not what you want.

Severity Code: 4.

AMDSADMP: OUTPUT DUMP DATA SET NAME IS INCORRECT. THE DATA SET NAME IS GREATER THAN 44 CHARACTERS.

Explanation: OUPTUT=(*Dunit,ddsname*) was specified, however, the data set name (ddsname) had a length greater than 44 characters.

System Action: Generation continues, however, no default dump data set name will be generated.

System Programmer Response: If a default dump data set name is desired, correct the OUTPUT= specification and regenerate the SADMP program.

Severity Code: 4.

AMDSADMP: OUTPUT DUMP DATA SET NAME IS INCORRECT. IT MUST CONTAIN THE TEXT ‘SADMP’.

Explanation: OUPTUT=(*Dunit,ddsname*) was specified, however, the data set name (ddsname) did not contain the text ‘SADMP’ as either part of, or as an entire data set qualifier.

System Action: Generation continues, however, no default dump data set name will be generated.

System Programmer Response: If a default dump data set name is desired, correct the OUTPUT= specification and regenerate the SADMP program.

Severity Code: 4.

AMDSADMP: REUSEDS=*reuseds* IS NOT ALLOWED. VALID SPECIFICATIONS ARE NEVER, CHOICE, OR ALWAYS. REUSEDS=CHOICE HAS BEEN USED.

Explanation: The REUSEDS operand is not NEVER, CHOICE, or ALWAYS.

System Action: Generation continues, using the default for the REUSEDS operand, CHOICE.

Severity Code: 0.

AMDSADMP: ULABEL=NOPURGE IS NOT POSSIBLE FOR A TAPE RESIDENCE VOLUME.

Explanation: The ULABEL cannot be NOPURGE when the IPL device is tape. SADMP ignores your ULABEL specification.

Severity Code: 8.

AMDSADMP: *keyword* IS AN OBSOLETE KEYWORD. IT IS IGNORED. SADMP GENERATION CONTINUES.

Explanation: An obsolete keyword is specified on the AMDSADMP macro. SADMP no longer requires the LOADPT or TYPE keywords to create a stand-alone dump program.

System Action: The system ignores the keyword and continues processing.

System Programmer Response: To eliminate this MNOTE, remove the indicated keyword and its associated parameter from the generation JCL.

Severity Code: 0.

AMDSADMP: ALIB=*alib* IS NOT VALID. THE REQUIRED SYNTAX IS ALIB=(VOLSER,UNIT).

Explanation: The system could not recognize the parameters specified on the ALIB keyword. The correct syntax is ALIB=(volser,unit), where volser is the volume serial number and unit is the UNIT=value of the device.

System Action: The system ignores this keyword and continues. The second step JCL might be incorrect.

System Programmer Response: Correct the syntax specified on the AMDSADMP macro and resubmit the JCL.

Severity Code: 8.

AMDSADMP: NUCLIB=*nuclib* IS NOT VALID. THE REQUIRED SYNTAX IS NUCLIB=(VOLSER,UNIT).

Explanation: The system could not recognize the parameters specified on the NUCLIB keyword. The correct syntax is NUCLIB=(volser,unit), where volser is the volume serial number and unit is the UNIT=value of the device.

System Action: The system ignores this keyword and continues. The second step JCL might be incorrect.

System Programmer Response: Correct the syntax specified on the AMDSADMP macro and resubmit the JCL.

Severity Code: 8.

AMDSADMP: MODLIB=*modlib* IS NOT VALID. THE REQUIRED SYNTAX IS MODLIB=(VOLSER,UNIT).

Explanation: The system could not recognize the parameters specified on the MODLIB keyword. The correct syntax is MODLIB=(volser,unit), where volser is the volume serial number and unit is the UNIT=value of the device.

System Action: The system ignores this keyword and continues. The second step JCL might be incorrect.

System Programmer Response: Correct the syntax specified on the AMDSADMP macro and resubmit the JCL.

Severity Code: 8.

AMDSADMP: LNKLIB=*lnklib* IS NOT VALID. THE REQUIRED SYNTAX IS MODLIB=(VOLSER,UNIT).

Explanation: The system could not recognize the parameters specified on the LNKLIB keyword. The correct syntax is LNKLIB=(volser,unit), where volser is the volume serial number and unit is the UNIT=value of the device.

System Action: The system ignores this keyword and continues. The second step JCL might be incorrect.

System Programmer Response: Correct the syntax specified on the AMDSADMP macro and resubmit the JCL.

Severity Code: 8.

AMDSADMP: CONSOLE TYPE *contp* IS INVALID. NO VALUE MAY BE SPECIFIED FOR SYSC. IT WILL BE IGNORED.

Explanation: A console type was specified following the console name of SYSC. No console type is allowed for this console.

System Action: The system ignores the specification.

System Programmer Response: None.

Severity Code: 0.

AMDSADMP: CONSOLE ADDRESS SYSC MAY ONLY BE SPECIFIED FOR THE FIRST CONSOLE. IT WILL BE IGNORED.

Explanation: The console name SYSC was not specified as the first console in the console list. SYSC may only be specified as the first console.

System Action: The system ignores the specification.

System Programmer Response: None.

Severity Code: 0.

AMDSADMP: ONLY SYSTEM CONSOLE DEFINED. DEFAULT (001F,3278) WILL ALSO BE USED.

Explanation: The console named SYSC was the only console that was defined. At least one 3270 console must also be defined.

System Action: The system defined a default console of (001F,3278).

System Programmer Response: None.

Severity Code: 0.

Coding the AMDSADMP Macro

This section describes the coding of the AMDSADMP macro, including the following topics:

- “Using the Dump Keyword to Request Additional Storage” on page 4-17
- “Dumping to a DASD Data Set” on page 4-22

Syntax of the AMDSADMP Macro

Figure 4-1 shows the AMDSADMP macro parameters.

```
[symbol] AMDSADMP

[,IPL={Tunit|Dunit|DSYSDA}]

 [,VOLSER={volser|SADUMP}]

  [,ULABEL={PURGE|NOPURGE}]

   [,CONSOLE=({cnum|(cnum,ctype) [,(cnum,ctype)]...|01F,3278}))]

    [,SYSUT={unit|SYSDA}]

     [,OUTPUT={Tunit|Dunit|(Dunit,ddsname)|T0282}]

      [,DUMP=('options')][,PROMPT]

       [,MSG={ACTION|ALLASIDS|ALL}]

        [,MINASID={ALL|PHYSIN}]

         [,COMPACT={YES|NO}]

          [,REUSEDS={CHOICE|ALWAYS|NEVER}]

           [,ALIB=(volser,unit)]

            [,NUCLIB=(volser,unit)]

             [,MODLIB=(volser,unit)]

              [,LNKLIB=(volser,unit)]

               [,DDSPROMPT={YES|NO}]
```

Figure 4-1. Format of AMDSADMP Macro Instruction

symbol
An arbitrary name you can assign to the AMDSADMP macro. stand-alone dump uses this symbol to create a job name for use in the initialization step.

AMDSADMP
The name of the macro.

IPL={Tunit|Dunit|DSYSDA}
Indicates the device number, device type, or esoteric name of the stand-alone dump residence volume. The first character indicates the volume type; T for tape, D for DASD.stand-alone dump uses the unit character string as the UNIT=value to allocate the residence volume for initialization.

A device number consists of 1 to 4 hexadecimal digits, optionally preceded by a slash (/). Use a slash preceding a 4-digit device number to distinguish it from a device type.

The default is IPL=DSYSDA. When you specify IPL=T, stand-alone dump assumes T3400. When you specify IPL=D, stand-alone dump assumes DSYSDA.

Notes:

1. This device will also contain a work file used during stand-alone dump processing.
2. This device will also contain a work file used during stand-alone dump processing.
3. It is not recommended to place the IPL text of stand-alone dump on a volume that contains page data sets. A restart of stand-alone dump (see under "Running the Stand-Alone Dump Program" on page 4-36) will hang during the real dump phase in this case.

VOLSER={volser|SADUMP}
Indicates the volume serial number the system is to use to allocate the residence volume for initialization. When you specify a tape volume, it must be NL (no labels). VOLSER=SADUMP is the default.

ULABEL={PURGE|NOPURGE}
Indicates whether stand-alone dump deletes (PURGE) or retains (NOPURGE) existing user labels on a DASD residence volume.When you specify NOPURGE, the stand-alone dump program is written on cylinder 0 track 0 of the residence volume, immediately following all user labels. If the user labels occupy so much space that the stand-alone dump program does not fit on track 0, the initialization program issues an error message and ends.

ULABEL=NOPURGE is the default.

CONSOLE=({cnum|(cnum,ctype)[,(cnum,ctype)]...|01F,3278})
Indicates the device numbers and device types of the stand-alone dump consoles that stand-alone dump is to use while taking the dump. When you specify CONSOLE=cnum, stand-alone dump assumes (cnum,3278). You can specify from two to 21 consoles by coding:

```
CONSOLE=((cnum,ctype),(cnum,ctype),[,(cnum,ctype)]...)
```

A device number consists of 3 or 4 hexadecimal digits, optionally preceded by a slash (/). Use a slash preceding a 4-digit device number to distinguish it from a device type.

The 3277, 3278, 3279, and 3290 device types are valid, and are interchangeable.

CONSOLE=(01F,3278) is the default.

You may specify CONSOLE=SYSC for the first console only. SYSC is a constant representing the hardware system console.

Note: The specification of CONSOLE does not affect the availability of the system console.

SYSUT={unit|SYSDA}
Specifies the UNIT=value of the device that stand-alone dump uses for work files during stand-alone dump initialization. You may specify the device as a

group name (for example, SYSDA), a device type (for example, 3330), or a unit address (for example, 131). SYSUT=SYSDA is the default.

OUTPUT={Tunit|Dunit|(Dunit,ddsname)|T0282}
Indicates the device type, number, and data set name that stand-alone dump uses as a default value if the operator uses the EXTERNAL INTERRUPT key to bypass console communication, or if the operator provides a null response to message AMD001A during stand-alone dump initialization. OUTPUT=T0282 is the default.

The device type can be specified as either a 'T' for tape or 'D' for DASD.

The device number consists of 3 or 4 hexadecimal digits, optionally preceded by a slash (/). Use a slash preceding a 4-digit device number to distinguish it from a device type.

If the default device is a DASD, you can also set up a default dump data set name to use by specifying both the device and the dump data set name on the OUTPUT= parameter. You may specify the first volume of a multi-volume DASD data set. If you specify a default dump data set name it must:

- Have a length that is 44 characters or less.
- Contain the text 'SADMP' as either part of, or as an entire data set qualifier.

Note that AMDSADMP processing does not allocate the data set or check to see that a valid MVS data set name has been provided. Therefore, the user should insure that:

- The AMDSADDD REXX utility is used to allocate and initialize the same data set name specified on the OUTPUT= keyword.
- The data set name specified should be fully qualified (without quotes).
- The necessary data set management steps are taken so that the stand-alone dump dump data sets will not be placed into a migrated state or moved to a different volume.
- Alphabetic characters appearing in the dump data set name should be specified as capital letters.

If the default DASD device is to be used and no dump data set name is provided, the stand-alone dump program will assume that the default dump data set name is SYS1.SADMP if the DDSPROMPT=NO parameter was also specified. Otherwise, if DDSPROMPT=YES was specified, the stand-alone dump program will prompt the operator at run-time for a dump data set name to use.

Notes:

1. At run-time, only a null response to message AMD001A will cause the stand-alone dump program to use the default device and/or dump data set name.
2. Do not place a data set that is intended to contain a stand-alone dump on a volume that also contains a page data set that the stand-alone dump program may need to dump. When stand-alone dump initializes a page volume for virtual dump processing, it checks to see if the output dump data set also exists on this volume. If it does, the stand-alone dump program issues message AMD100I and does not retrieve any data from page data sets on this volume. Thus, the dump may not contain all of the data that you requested. This lack of data may impair subsequent diagnosis.
3. You cannot direct output to the stand-alone dump residence volume.

DUMP='*options*'
Indicates additional virtual storage that you want dumped. This storage is described as address ranges, dataspaces, and subpools in address spaces. When you do not specify DUMP, stand-alone dump does not dump any additional storage unless you specify PROMPT. See "Using the Dump Keyword to Request Additional Storage" on page 4-17 for more information.

PROMPT
Causes stand-alone dump, at run time, to prompt the operator for additional virtual storage to be dumped. The operator can respond with the same information that can be specified for the DUMP keyword. When you do not specify PROMPT, stand-alone dump does not prompt the operator to specify additional storage. See "Using the Dump Keyword to Request Additional Storage" on page 4-17 for more information.

MSG={ACTION|ALLASIDS|ALL}
Indicates the type of stand-alone dump messages that appear on the console. When you specify ACTION, stand-alone dump writes only messages that require operator action. When you specify ALL, stand-alone dump writes most messages to the console. However, messages AMD010I, AMD057I, AMD076I, AMD081I, and AMD102I appear only in the stand-alone dump message log. When you specify ALLASIDS, the stand-alone dump program behaves as if MSG=ALL was specified, except that message AMD010I also appears on the console. ALL is the default.

This keyword has no effect on the stand-alone dump message log; even if you specify MSG=ACTION, the stand-alone dump virtual dump program writes all messages to the message log in the dump.

MINASID={ALL|PHYSIN}
Indicates the status of the address spaces that are to be included in the minimal dump. Specify PHYSIN to dump the minimum virtual storage (LSQA and selected system subpools) for the physically swapped-in address spaces only. Specify ALL to dump the minimum virtual storage (LSQA and selected system subpools) for all of the address spaces. ALL is the default.

At run time, if PHYSIN was specified, stand-alone dump writes message AMD082I to the operator's console to warn the operator that some virtual storage might be excluded from the dump.

COMPACT={YES|NO}
COMPACT(YES) compacts the data stored on a tape cartridge if theIDRC hardware feature is available on your tape drive. If the IDRC feature is available and you do not specify the COMPACT keyword, the default is YES, so that IDRC will compact the dump data. Otherwise, the data is handled as usual.

REUSEDS={CHOICE|ALWAYS|NEVER}
Indicates whether stand-alone dump should reuse the dump data set on the specified output device when it determines that the data set is valid, however, it may contain data from a previous dump. Stand-alone dump determines this by checking to see if the first record in the data set matches the record that is written by the AMDSADDD rexx utility. When you specify ALWAYS, stand-alone dump issues message AMD094I and reuses the specified dump data set. When you specify NEVER, stand-alone dump issues message AMD093I and prompts the operator, through message AMD001A, for an output device. When you specify CHOICE, stand-alone dump informs the operator, with message AMD096A, that the data set is not reinitialized and requests permission to reuse the data set. See for more information about defining, clearing, and reallocating the dump data set.

CHOICE is the default.

ALIB=(volser,unit)
Specifies the volume serial number and UNIT=value of the volume that contains all of the following system data sets:

- SYS1.MODGEN
- SYS1.LINKLIB
- SYS1.NUCLEUS

This parameter is valid only when you are generating the stand-alone dump program using two-stage generation.

Note: The specification of the NUCLIB, LNKLIB, or MODLIB parameters overrides the corresponding value specified on the ALIB parameter.

See “Using Two-Stage Generation when Migrating” on page 4-33 for information on the use of this parameter.

NUCLIB=(volser,unit)
Specifies the volume serial number and UNIT=value of the volume that contains the system data set SYS1.NUCLEUS. This parameter is valid only when you generate the stand-alone dump program using two-stage generation.

See “Using Two-Stage Generation when Migrating” on page 4-33 for information on the use of this parameter.

MODLIB=(volser,unit)
Specifies the volume serial number and UNIT=value of the volume that contains the system data set SYS1.MODGEN. This parameter is valid only when you generate the stand-alone dump program using two-stage generation.

See “Using Two-Stage Generation when Migrating” on page 4-33 for information on the use of this parameter.

LNKLIB=(volser,unit)
Specifies the volume serial number and UNIT=value of the volume that contains the system data set SYS1.LINKLIB. This parameter is valid only when you generate the stand-alone dump program using two-stage generation.

See “Using Two-Stage Generation when Migrating” on page 4-33 for information on the use of this parameter.

DDSPROMPT={YES|NO}
DDSPROMPT=YES allows the stand-alone dump program to prompt the operator for an output dump data set when dumping to a DASD device. When DDSPROMPT=YES is specified, after replying to message AMD001A with a DASD device number, message AMD002A will also be issued to prompt the operator for a dump data set name.

DDSPROMPT=NO indicates that the stand-alone dump program should not prompt for a dump data set name when dumping to a DASD device. When DDSPROMPT=NO is specified, after replying to message AMD001A with a DASD device number, the stand-alone dump program will assume that the data set SYS1.SADMP is to be used. DDSPROMPT=NO is the default.

Note that regardless of the DDSPROMPT= keyword value, you can always use a default device and dump data set name by specifying the OUTPUT=(Dunit,ddsname) keyword. The stand-alone dump program uses the default values specified on the OUTPUT= keyword when the operator uses the

EXTERNAL INTERRUPT key to bypass console communication, or if the operator provides a null response to message AMD001A.

Examples of Coding the AMDSADMP Macro

The following examples show how to code the AMDSADMP macro to create various kinds of stand-alone dump programs.

Example: Accepting All Defaults

This example shows the AMDSADMP macro coded without explicitly specified parameters to generate a direct access resident dump program.

```
DUMP1 AMDSADMP
```

The defaults are:

```
IPL=DSYSDA
VOLSER=SADUMP
ULABEL=NOPURGE
CONSOLE=(01F,3278)
SYSUT=SYSDA
OUTPUT=T282
MSG=ALL
MINASID=ALL
COMPACT=YES
REUSEDS=CHOICE
DDSPROMPT=NO
```

Example: Generating an Unformatted, Tape Resident Dump Program

In this example, the IPL parameter specifies tape as the residence volume, and the VOLSER parameter identifies that tape. All other parameters are allowed to default.

```
AMDSADMP IPL=T3400,VOLSER=SATAPE
```

The defaults are:

```
ULABEL=NOPURGE
CONSOLE=(01F,3278)
SYSUT=SYSDA
OUTPUT=T282
MSG=ALL
MINASID=ALL
COMPACT=YES
REUSEDS=CHOICE
DDSPROMPT=NO
```

Example: Generating a Dump Program with Output to DASD

In this example, the OUTPUT parameter directs the stand-alone dump output to dump data set SYS1.SADMP on device 450, and the REUSEDS parameter specifies that the operator will be prompted on whether to reuse the dump data set.

```
AMDSADMP OUTPUT=D450,REUSEDS=CHOICE
```

The defaults are:

```
IPL=DSYSDA
VOLSER=SADUMP
ULABEL=NOPURGE
CONSOLE=(01F,3278)
SYSUT=SYSDA
MSG=ALL
MINASID=ALL
COMPACT=YES
DDSPROMPT=NO
```

Example: Generating a Dump Program with Output to DASD

In this example, the OUTPUT parameter directs the stand-alone dump output to dump data set SADMP.DDS1 on device 450. Furthermore, the DDSPROMPT=YES keyword allows for run-time dump data set prompting.

```
AMDSADMP OUTPUT=(D450,SADMP.DDS1),DDSPROMPT=YES
```

The defaults are:

```
IPL=DSYSDA
VOLSER=SADUMP
ULABEL=NOPURGE
CONSOLE=(01F,3278)
SYSUT=SYSDA
MSG=ALL
MINASID=ALL
COMPACT=YES
REUSEDS=CHOICE
```

Recommended specification during build process:

```
SP(ALL) IN ASID(1,'JESXCF')
ALSO DATASPACES OF ASID(1,'JESXCF', 'APPC', 'SMSVSAM', 'CONSOLE', 'SYSBMAS')
ALSO PAGETABLES OF DATASPACES
```

If you run JES2, add:

```
ALSO SP(ALL) IN ASID('JES2')
```

Additional subpools and dataspaces may be needed, depending on your installed IBM, vendor, and locally-written products and applications.

Using the Dump Keyword to Request Additional Storage

You can request that stand-alone dump dump additional storage in two ways:

- **Specifying Dump Options on the AMDSADMP Macro**

Specify the dump tailoring options described in "Dump Tailoring Options" on page 4-19 within parentheses and single quotes as the value of the DUMP keyword on the AMDSADMP macro.

Examples:

```
DUMP=('SP(5,37,18) IN ASID('JES3')')
DUMP=('RANGE(0:1000000) IN ASID(1)')
DUMP=('DATASPACES OF ASID('DUMPSRV')')
```

Note: Do not double the quotes within the DUMP options. The DUMP options cannot exceed 255 characters in length.

- **Specifying Additional Dump Options at Run Time**

  By coding the PROMPT keyword on the AMDSADMP macro, you can have Stand-alone dump prompt the operator to dump additional storage. When you code PROMPT, and the virtual storage dump program gets control, stand-alone dump issues the following message:

  ```
  AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
  ```

  The operator can respond with one of the following:

  – DUMP followed by dump options. In this case, the '=' after DUMP is optional. See "Dump Tailoring Options" on page 4-19 for the possible dump options.

  – SET followed by the MINASID options.

  – LIST. On the console, stand-alone dump displays the current virtual storage areas to be dumped.

  – END. Stand-alone dump stops prompting the operator for options and begins processing.

  Figure 4-2 shows a sample exchange between stand-alone dump and the operator. The operator's replies are in lowercase. Note the operator's reply to message AMD059D using the DUMP keyword.

```
 AMD082I WARNING: THE MINASID SPECIFICATION HAS BEEN SET TO 'PHYSIN'.
 AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
     dump sp(0::9) inasid('jes2')
 AMD060I ERROR IN INPUT TEXT INDICATED BY '*':
 DUMP SP(0::9) INASID('JES2')
            *
AMD065A ENTER TEXT TO BE SUBSTITUTED FOR THE TEXT IN ERROR.
 >
 AMD060I ERROR IN INPUT TEXT INDICATED BY '*':
 DUMP SP(0:9) INASID('JES2')
               ******
 AMD065A ENTER TEXT TO BE SUBSTITUTED FOR THE TEXT IN ERROR.
 > in asid
 AMD082I WARNING: THE MINASID SPECIFICATION HAS BEEN SET TO 'PHYSIN'.
 AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
 > list
 AMD067I CURRENT DUMP OPTIONS:
   CSA ALSO LSQA, SP(203:205,213:215,229:230,236:237,247:249) IN ASID(PHYSIN)
   ALSO SP(0:9) IN ASID('JES2')
 AMD082I WARNING: THE MINASID SPECIFICATION HAS BEEN SET TO 'PHYSIN'.
 AMD059D ENTER 'DUMP' OR 'SET' WITH OPTIONS, 'LIST' OR 'END'.
 > end
```

Figure 4-2. Sample Console Output from the Stand-Alone Dump Program

When stand-alone dump detects an error in the reply to message AMD059D, it repeats the incorrect line at the console, underscores the incorrect part with asterisks, and prompts the operator for replacement text. If the dump options exceed 255 characters, stand-alone dump marks the whole line in error.

If a system restart occurs during the virtual storage dump program, stand-alone dump re-prompts the operator for dump options. stand-alone dump does not use any of the dump options that the operator specified before the system restart.

Dump Tailoring Options: You can specify the dump tailoring options in one or both of the following ways:
- On the DUMP keyword of the AMDSADMP macro
- By the operator in reply to message AMD059D at run time.

Following is a list of the dump tailoring options you can specify. For a complete explanation of the options, see "Explanation of Dump Tailoring Options" on page 4-20.

```
{dump-spec-list|SET MINASID{ALL|PHYSIN}|LIST|END}
```

dump-spec-list is one or more of the following:
- *range-spec-list* IN ASID(*address-space-list*) [ALSO...]
- DATASPACES OF *domain-spec-list*[...]
- DSP OF *domain-spec-list*[...]
- PAGETABLES OF DATASPACES

  range-spec-list is one or more of the following:
  - SP(*subpool-list*)
  - RANGE(*address-range-list*)
  - LSQA
  - HIGH VIRTUAL

    subpool-list is one of the following:
    - *subpool-number* TO *subpool-number* [,...]
    - ALL

    address-range-list is one of the following:
    - *address* TO *address* [,...]
    - ALL

  address-space-list is one of the following:
  - *asid* TO {*asid*|*jobname*|SYSKEY|PHYSIN}[,...]
  - ALL

Use the following guidelines when specifying values in your dump tailoring options:
- *address* is a hexadecimal number from 0 to X'7FFFFFFF'
- *subpool-number* is a decimal number from 0 to 255
- *asid* is a hexadecimal number from 0 to X'FFFF'
- *jobname* is a valid jobname enclosed in single quotes. Including wildcard characters is valid for jobnames. For a description of wildcard characters, see the ASID('jjjj') option in the topic "Explanation of Dump Tailoring Options" on page 4-20.
- *range-spec-list* is a list of subpools, a list of storage ranges, or both
- *domain-spec-list* is a list of address spaces
- 'TO' and ':' are synonyms
- 'DATASPACES' and 'DSP' are synonyms

Keywords, such as DATASPACES, can be truncated on the right, provided the truncated form is not ambiguous. You may enter letters in either lower-case or uppercase. Blanks can be inserted between numbers, keywords, and separators; blanks cannot be inserted within numbers or keywords.

Explanation of Dump Tailoring Options: This section provides an explanation for each of the dump tailoring options.

RANGE(*xxxxxxxx:yyyyyyyy,xxxxxxxx:yyyyyyyy...***)**
Specifies one or more ranges of storage that you want dumped. *xxx* and *yyy* are hexadecimal addresses from 0 to X'7FFFFFFF'

RANGE(ALL)
Specifies dumping of all storage from 0 to X'7FFFFFFF'

SP(*ddd***)**
Causes stand-alone dump to dump subpool *ddd*. *ddd* is a decimal integer from 0 to 255.

SP(*ddd***:***eee***)**
Causes stand-alone dump to dump all subpools from *ddd* to *eee*, inclusive.

SP(*ddd***:***eee***,***ddd***:***eee,...***)**
Causes stand-alone dump to dump the combination of subpools that you specify.

SP(ALL)
Causes stand-alone dump to dump all subpools, from 0 to 255 inclusive.

LSQA
Causes stand-alone dump to dump the LSQA.

HIGH VIRTUAL
Causes stand-alone dump to dump all allocated storage above 2G.

ASID(*xxxx***:***yyyy***)**
Causes stand-alone dump to dump storage for the range of address spaces whose ASIDs begin at *xxx* and end at *yyy*, inclusive. *xxx* and *yyy* are hexadecimal numbers from X'1' to X'FFFF'.

ASID('*jjj***')**
Causes stand-alone dump to dump storage for the address space that jobname *jjj* identifies. Note that you must enclose the jobname in single quotes.

Wildcard Characters: You can use wildcard characters to identify multiple jobnames. The valid wildcard characters are:

***** Zero or more characters, up to the maximum length of the string. An asterisk can start the string, end it, appear in the middle of the string, or appear in several places in the string. A single asterisk for the jobname indicates that all jobnames will match.

? One character. One or more question marks can start the string, end it, appear in the middle of the string, or appear in several places in the string. A single question mark indicates all jobnames consisting of one character.

ASID(SYSKEY)
Causes stand-alone dump to dump storage for all address spaces whose active TCB has an associated storage key of 0 to 7.

ASID(*combination***)**
You may combine any of the above specifications. An example of a valid combination is ASID(2,'IMSJOB',SYSKEY).

ASID(PHYSIN)
Causes stand-alone dump to dump storage for physically swapped-in address spaces.

ASID(ALL)
Causes stand-alone dump to dump storage for all address spaces. Note that you cannot specify ASID(ALL) in combination with any of the other ASID specifications.

DATASPACES OF ASID(*qualifier*)
When you specify the `DATASPACES OF ASID(qualifier)` keyword, stand-alone dump dumps all data spaces owned by the specified address space. For each requested data space, stand-alone dump:

- Dumps pages backed by central storage during the central storage dump
- Copies into central storage and dumps every page that is not a first reference page and not backed by central storage

PAGETABLES OF DATASPACES
When you specify the `PAGETABLES OF DATASPACES` keyword, stand-alone dump dumps paged-out virtual storage that contains the page tables for all data spaces.

When stand-alone dump dumps the storage that you specify, stand-alone dump dumps all listed subpools and address ranges in all specified address spaces for each specification of dump options. However, stand-alone dump does not merge your specifications across the dump options that you specify. For example, to cause stand-alone dump to dump subpools 0 and 1 in address space A, and subpools 0 and 1 in address space B, enter:

```
DUMP SP(0,1) IN ASID(A,B)
```

To cause stand-alone dump to dump subpool 0 in address space A and subpool 1 in address space B, enter:

```
DUMP SP(0) IN ASID(A) ALSO SP(1) IN ASID(B)
```

The following are examples of valid specifications:

```
DUMP SP(0:7,15),RANGE(0:10000000) IN ASID(SYSKEY),ASID(8)
DU  (SP(0 TO 7 OR 15),SP(255)) IN AS('TCAM')
DUMP RANGE(ALL) IN ASID(1) ALSO SP(0) IN ASID(SYSKEY,8)
DU DAT OF AS(ALL)
DUMP ( SP(0:127) IN ASID('GENER') ALSO SP(0) IN ASID('IMS') )
DUMP LSQA IN AS('MYJOB',14)
DU SP(128),LS IN ASID(C,PHYSIN)
DUMP DATASPACES OF ASID('MYJOB??')
DUMP DATASPACES OF ASID('MY*')
DUMP HIGH VIRTUAL IN ASID(C)
```

Specifying the Minimal Stand-Alone Dump: Stand-alone dump is designed to produce a minimal dump that includes certain system-related storage ranges in all address spaces, as well as dataspaces associated with certain address spaces.

The Stand-Alone dump user can request additional storage to be dumped by coding PROMPT on the AMDSADMP macro, and responding to message AMD059D, or by coding the DUMP keyword on the AMDSADMP macro. (See “Using the Dump Keyword to Request Additional Storage” on page 4-17 for more information on this topic.)

The minimal dump is requested with the MINASID keyword on the AMDSADMP macro. MINASID=ALL produces a dump of:

- all of CSA and LSQA
- subpools 203-205, 213-215, 229-230, 236-237 and 247-249 in all address spaces
- the full GTF address space
- subpool 127 in the GRS address space
- dataspaces whose names begin with ISG for the GRS address space
- all of DUMPSRV's dataspaces
- the full XCF address space
- all of XCF's dataspaces
- XES-related dataspaces for address spaces with an XES connection

MINASID=PHYSIN produces a dump of the same areas, but restricts the address spaces dumped to those which are physically swapped-in at the time the system writes the dump. If MINASID is not specified on the AMDSADMP macro, the default of MINASID=ALL is assumed. You can also reply SET MINASID (PHYSIN|ALL) to message AMD059D if you coded PROMPT on the AMDSADMP macro.

Example: Minimal Dump Option Specified at Stand-Alone Dump Generation

Use the following to restrict the minimal dump option to include only physically swapped-in address spaces, during stand-alone dump generation.

```
AMDSADMP IPL=SYSDA,VOLSER=VSSA02,MINASID=PHYSIN
```

Example: Minimal Dump Option Specified at Run-time

Reply to message AMD059D with the following SET command to request that the minimal dump option include only physically swapped-in address spaces, while stand-alone dump is running.

```
SET MINASID(PHYSIN)
```

Dumping to a DASD Data Set

When you specify DASD on the OUTPUT parameter, you direct the output of the stand-alone dump program to a predefined dump data set on one of the following types of DASD:

- 3380
- 3390
- 9345

Note: The selection of the output device (DASD or tape) can be made at both generation time and at run time. An output device specified at run time overrides an output device specified at generation time.

When preparing to take a stand-alone dump to DASD, you must allocate and initialize the dump data set using the REXX utility AMDSADDD. See for more information.

The following requirements exist for the allocation of the DASD dump data set:

- The dump data set must have the text 'SADMP' as either part of, or as an entire data set qualifier.

- Do not place a data set that is intended to contain a stand-alone dump on a volume that also contains a page data set that the stand-alone dump program you may need to dump. When stand-alone dump initializes a page volume for virtual dump processing, it checks to see if the output dump data set also exists on this volume. If it does, the stand-alone dump program issues message AMD100I and does not retrieve any data from page data sets on this volume. Thus, the dump may not contain all of the data that you requested. This lack of data may impair subsequent diagnosis.
- The dump data set cannot be defined on the same volume that contains the IPL text of stand-alone dump.

  Note: Because the data set does not have to be cataloged, there can be more than one dump data set with the same name per system. Furthermore, because the data set can be uniquely named, there can be more than one dump data set per volume.
- IBM recommends that you define the dump data set on a volume that does not contain any other data sets, especially volumes that contain sysplex couple data sets. This will ensure maximum capacity when needed and avoid the possibility of other data sets being accessed by another system.
- The dump data set must be both allocated and initialized using the stand-alone dump REXX utility AMDSADDD. See for more information.
- Since the stand-alone dump program must be able to locate the dump data set on the output device being used, it is imperative that the necessary data set management steps be taken so that the stand-alone dump dump data sets will not be placed into a migrated state or moved to a different volume. The dump data sets must also be exempt from any space management processing that will release unused space.

When the dump data set is filled, the stand-alone dump program prompts the operator, with message AMD001A, to specify another output device. The stand-alone dump program can continue dumping to any stand-alone dump supported device, however, once a tape device is selected, it must be used to complete the dump even though multiple tape volumes may be required.

Note: Dumping to multiple DASD dump data sets implies that each dump data set used has been preformatted by the AMDSADDD REXX utility.

Using the AMDSADDD Utility

The REXX utility AMDSADDD resides in SYS1.SAMPLIB. To use the utility, either copy AMDSADDD from SYS1.SAMPLIB into your installation's REXX library and run the utility from there or run the utility directly from SYS1.SAMPLIB. This section describes how to use the AMDSADDD REXX utility to:

- Allocate and initialize the data set. See Figure 4-3 on page 4-26 for an example of allocating and initializing the dump data set.
- Clear (reinitialize) the data set. See Figure 4-4 on page 4-27 for an example of clearing the dump data set.
- Reallocate and initialize the data set. See Figure 4-5 on page 4-27 for an example of reallocating and initializing the dump data set

The data set allocated by the AMDSADDD REXX utility must have these characteristics:

- The data set name (DSNAME) must:
  - be 44 characters or less in length
  - contain the text 'SADMP' as either part of, or as an entire data set qualifier.

For example, valid dump data set names are:
- SYS1.SADMP
- SADMP
- SYSTEMA.SADMPDDS

Invalid dump data set names are:
- SYS1.DUMP.DATASET
- SADUMP

- The block size (BLKSIZE) must be 20800 for a 3380 or 9345 DASD, and 24960 for a 3390 DASD.
- The logical record length (LRECL) must be 4160.
- The record format (RECFM) must be FBS.
- The data set organization (DSORG) must be PS.
- The data set must consist of a single extent.

Note: Stand-alone dump processing can use a 3390 DASD defined with a block size of 20800; however, the allocated space will not be fully utilized unless a block size of 24960 is used. The AMDSADDD REXX utility will allocate 3390 DASD devices using a block size of 24960.

You provide the volume, dump data set name, unit, space, and catalog disposition on the invocation of the AMDSADDD REXX utility. If multiple volumes are specified, then a multi-volume data set will be allocated and formatted. Up to 16 volumes may be specified, all having the same device type. The amount of space specified for the data set will be allocated on each volume.

Special control information is written to multi-volume data sets to allow all of the volumes to be located when the data set is written to. This includes the device number of the volume. The data set will not be usable by stand-alone dump if the control information is missing or invalid. If a volume of a multi-volume data set is moved to a new device number, the data set must be re-initialized to update the control information. The data set cannot be used by a system that has the volumes attached at a device number different than the system which writes the control information.

When using multi-volume data sets, it is highly recommended that they be cataloged. This simplifies processing, as IPCS can easily be used to format and copy the dump data in the cataloged data sets.

Note: REXX requires that the specified parameters appear in the order listed. If you do not specify a parameter, the AMDSADDD REXX utility prompts for a specification of that parameter.

```
AMDSADDD
  {DEFINE|CLEAR|REALLOC} volser{(data set name)} type [space] [YES NO]
```

or

```
AMDSADDD
  {DEFINE|CLEAR|REALLOC} (volumelist){(data set name)} type [space] [YES NO]
```

AMDSADDD
The name of the REXX utility.

DEFINE|CLEAR|REALLOC
Indicates the function to be performed by the AMDSADDD REXX utility:

DEFINE Allocates and initializes a new dump data set.

CLEAR Reinitializes an existing dump data set. Once cleared, the data set is ready for use.

REALLOC Deletes an existing SYS1.SADMP dump data set, then reallocates and reinitializes a new SYS1.SADMP dump data set on the same volume. Use REALLOC, for example, to increase the size of the dump data set. If the existing dump data set does not exist, AMDSADDD will convert the function to a DEFINE request and continue using DEFINE processing. Use REALLOC, for example, to increase the size of the dump data set. If the request to reallocate and reinitialize a new SYS1.SADMP dump data set cannot be satisfied (for example, if the user attempts to reallocate SYS1.SADMP using more cylinders than are available), AMDSADDD may delete the existing SYS1.SADMP dump data set

volser{(data set name)}
Indicates the VOL=SER= name of the volume on which the dump data set is to be allocated. Do not use the stand-alone dump residence volume or the volumes containing the system paging data sets.

Optionally, also defines the dump data set name to be allocated on the volume. If data set name is specified, it must:

- be fully qualified (without quotes)
- have a length of 44 characters or less
- contain the text ’SADMP’ as either part of, or as an entire data set qualifier.

Note: If no data set name is specified, the AMDSADDD utility will allocate the data set SYS1.SADMP on the specified volume.

(vollist){(data set name)}
vollist is a comma delineated list of volsers to use for the data set. A multi-volume data set will be allocated using the list of volumes. The device number of the first volume is used to specify the data set to stand-alone dump.

type
Indicates the device type on which the dump data set should be allocated. Valid DASD types are 3380, 3390, and 9345.

space
Indicates the number of cylinders for the dump data set to be allocated. For a multi-volume data set, this amount is allocated on each volume.

The size of your dump output depends on your storage configuration and how much of that storage you choose to dump using the options of stand-alone dump. To estimate how much space, in cylinders, to allocate for your dump data set, use the number of cylinders of DASD that a typical dump to tape consumes when it has been copied to DASD for IPCS processing. If you do not allocate enough space, the stand-alone dump program prompts the operator, through message AMD001A and message AMD002A (if DDSPROMPT=YES was specified on the AMDSADMP macro), to specify a different device and/or a different dump data set so that dumping can continue.

The *space* option is not required with the CLEAR parameter. The *space* option is, however, required with the DEFINE and REALLOC parameters.

YES|NO
Specifies whether the system is to catalog the dump data set. If you want the

data set to be cataloged, specify **YES** or **Y**. If you do not want the data set to be cataloged, specify **NO** or **N**. Specifying **N** allows you to allocate multiple dump data sets with the same name.

The catalog option is not required with the CLEAR parameter. The catalog option is, however, required with the DEFINE and REALLOC parameters.

Figure 4-3 shows an example of using the AMDSADDD REXX utility to allocate and initialize the dump data set with a size of 350 cylinders and a VOL=SER= of SAMPLE. Since no data set name is specified, AMDSADDD allocates the dump data set SYS1.SADMP on the volume SAMPLE.

Note: Stand-alone dump does not issue error messages during the processing of AMDSADDD. Stand-alone dump does, however, pass messages to the operator from other sources, such as the TSO/E ALLOC command.

```
 ------------------------  TSO COMMAND PROCESSOR  -------------------
 ENTER TSO COMMAND, CLIST, OR REXX EXEC BELOW:
 ===> exec rexx.exec(amdsaddd)
 What function do you want?
 Please enter DEFINE if you want to allocate a new dump dataset
 Please enter CLEAR if you want to clear an existing dump dataset
 Please enter REALLOC if you want to reallocate and clear an existing
        dump dataset
 Please enter QUIT if you want to leave this procedure
define
 Please enter VOLSER or VOLSER(dump_dataset_name)
sample
 Please enter the device type for the dump dataset
 Device type choices are 3380 or 3390 or 9345
3380
 Please enter the number of cylinders
350
 Do you want the dump dataset to be cataloged?
 Please respond Y or N
y
 IKJ56650I TIME-11:00:00 PM. CPU-00:00:00 SERVICE-20191 SESSION-00:09:55 JUNE
14,1994
Initializing output dump dataset with a null record:
Dump dataset has been successfully initialized

Results of the DEFINE request:

  Dump Dataset Name    : SYS1.SADMP
  Volume               : SAMPLE
  Device Type          : 3380
  Allocated Amount     : 350
```

Figure 4-3. Using AMDSADDD to Allocate and Initialize a Dump Data Set

Figure 4-4 on page 4-27 shows an example of using the AMDSADDD REXX utility to clear (reinitialize) an existing dump data set called SADMP.DDS1 on VOL=SER=SAMPLE. In this example, the parameters are part of the invocation of the utility; therefore, AMDSADDD does not prompt for values.

```
 ------------------------  TSO COMMAND PROCESSOR  --------------------
 ENTER TSO COMMAND, CLIST, OR REXX EXEC BELOW:
 ===> exec rexx.exec(amdsaddd) 'clear sample(sadmp.dds1) 3380'
IKJ56650I TIME-11:00:00 PM. CPU-00:00:00 SERVICE-20191 SESSION-00:09:55 JUNE
14,1994
Initializing output dump dataset with a null record:
Dump dataset has been successfully initialized
Results of the CLEAR request:

  Dump Dataset Name    : SADMP.DDS1
  Volume               : SAMPLE
  Device Type          : 3380
  Allocated Amount     : 350

***
```

Figure 4-4. Using AMDSADDD to Clear an Existing Dump Data Set

Figure 4-5 shows an example of using the AMDSADDD REXX utility to delete an existing dump data set and allocate a new dump data set called SYSTEM1.SADMPDDS on VOL=SER=SMS001. In this example, the parameters are part of the invocation of the utility; therefore, AMDSADDD does not prompt for values.

Note: In this example, AMDSADDD has indicated that the dump data set has been cataloged on a different volume than was requested. This scenario is possible if the dump data set is allocated in an SMS environment.

```
 ------------------------  TSO COMMAND PROCESSOR  --------------------
 ENTER TSO COMMAND, CLIST, OR REXX EXEC BELOW:
 ===> exec rexx.exec(amdsaddd) 'realloc SMS001(SYSTEM1.SADMPDDS) 3390 100 Y'
IKJ56650I TIME-11:00:00 PM. CPU-00:00:00 SERVICE-20191 SESSION-00:09:55 JUNE
14,1994
Output dump dataset is being allocated on volume SMS002

Initializing output dump dataset with a null record:
Dump dataset has been successfully initialized

Results of the REALLOC request:

  Dump Dataset Name    : SYSTEM1.SADMPDDS
  Volume               : SMS002
  Device Type          : 3390
  Allocated Amount     : 100

***
```

Figure 4-5. Using AMDSADDD to Reallocate the Dump Data Set

Example: Running AMDSADDD in Batch Mode

Use this JCL to allocate and initialize the dump data set SADMP.DDS2 on VOL=SER=USRD53 with a size of 2653 cylinders.

Note: Because users cannot be prompted to enter values when invoking the AMDSADDD REXX utility in batch mode, you must specify all parameters in the order listed.

```
//SAMPLE  JOB 'S3031,B7100003,S=C','BATCH EXAMPLE',RD=R,
//          MSGLEVEL=(1,1),CLASS=E,NOTIFY=SAMPLE,MSGCLASS=H
//STEP1    EXEC  PGM=IKJEFT01,REGION=64M
//SYSTSPRT   DD  SYSOUT=*
//SYSTSIN    DD  *
EXEC 'SAMPLE.AMDSADDD.EXEC' 'DEFINE USRD53(SADMP.DDS2) 3380 2653 N'
/*
```

Generating the Stand-Alone Dump Program

After coding the AMDSADMP macro, you can generate the stand-alone dump program. There are two ways to generate the stand-alone dump program:
- One-step generation
- Two-stage generation

IBM recommends that you use one-step generation to generate the stand-alone dump program because multiple tasks are performed in one step. Two-stage generation is useful for the following purposes:
- Generating multiple stand-alone dump programs simultaneously.
- Generating a new version of the stand-alone dump program when migrating to a new version of MVS.
- Simplest to use when you must override to use a different systems database to
  - Assemble macro to generate JCL
  - Run JCL

One-Step Generation

In one-step generation, run the AMDSAOSG program as a single job step, using the AMDSADMP macro you have coded as input data on the SYSIN control statement.

Example: One-Step Generation with Overriding DDNAMES

The stand-alone dump utility program, AMDSAOSG, initializes a stand-alone dump residence volume in one job step by dynamically allocating data sets and invoking the appropriate programs. To run the one-step generation program, indicate one AMDSADMP macro as a control statement for DDNAME GENPARMS.

```
//SADMPGEN JOB  MSGLEVEL=(1,1)
//OSG      EXEC PGM=AMDSAOSG
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR
//         DD   DSN=SYS1.MODGEN,DISP=SHR
//GENPRINT DD   DSN=SADMP.LIST,DISP=OLD
//GENPARMS DD   *
         AMDSADMP IPL=DSYSDA,VOLSER=SPOOL2,                            X
               CONSOLE=(1A0,3277)
         END
/*
```

Table 4-1 (which has related notes that follow it) shows the DDNAMES AMDSAOSG uses, and the defaults for the DDNAMES.

Table 4-1. DDNAMES and Defaults Used by AMDSAOSG

ddname	Default Value	Use
DPLTEXT	DSN=SYS1.NUCLEUS(AMDSADPL),DISP=SHR	Input for AMDSABLD.
DVITEXT	DSN=SYS1.NUCLEUS(AMDSADVI),DISP=SHR	Input for AMDSABLD.
GENPARMS	Must be preallocated.	Input for AMDSAOSG, passed to assembler.
GENPRINT	SYSOUT=A	Output listing from AMDSAOSG.
IPITEXT	DSN=SYS1.NUCLEUS(AMDSAIPI),DISP=SHR	Input for AMDSABLD.
IPLDEV	DSN=SYS1.PAGEDUMP.Vvolser,UNIT=iplunit, VOL=(PRIVATE,SER=iplser),	Stand-alone dump program, output from AMDSABLD. ICKDSF uses VOL keywords to describe the residence volume.
	DISP=OLD,DCB=(BLKSIZE=12288,RECFM=U, DSORG=PS), LABEL=(,NL)	Tape IPL volume.
	DISP=(NEW,KEEP),DCB=(LRECL=4096,BLKSIZE=4096, RECFM=F,DSORG=PS),SPACE=(4096,(1095),,CONTIG), LABEL=EXPDT=99366	DASD IPL volume.
IPLTEXT	DSN=SYS1.NUCLEUS(AMDSAIPD),DISP=SHR for DASD	Input for AMDSABLD.
	DSN=SYS1.NUCLEUS(AMDSAIPT),DISP=SHR for tape	
PGETEXT	DSN=SYS1.NUCLEUS(AMDSAPGE),DISP=SHR	Input for AMDSABLD.
SYSPRINT	Must not be preallocated	Temporary listings from called programs.
SYSPUNCH	DSN=&OBJ,UNIT=SYSDA,SPACE=(80,(250,50))	Object module passed from assembler to AMDSABLD.
SYSTERM	None	Assembly messages.
SYSUT1	UNIT=SYSDA,SPACE=(1700,(400,50))	Work file for assembler.
TRK0TEXT	DSN=&TRK0TEXT,UNIT=iplunit, VOL=SER=iplser,SPACE=(4096,(2,1))	Cylinder 0, Track 0 IPL text from AMDSABLD to ICKDSF.

Notes:
1. You **must** specify the GENPARMS DDNAME on the job step.
2. You **cannot** specify the SYSPRINT and SYSIN DD statements in the job step.
3. In GENPARMS, you specify values for UNIT= and VOLSER= on the AMDSADMP macro statement.

Example: One-Step Generation of Stand-Alone Dump to DASD

This JCL generates a stand-alone dump from DASD 222 using a volume serial of SADMPM. The output will be directed to the data set SYS1.SADMP on DASD 450. Stand-alone dump determines at run-time if that device is usable. If the dump data set on device 450 is not usable, the operator will be prompted for another data set. The operator can press enter on any of the consoles at address 041, 042, 0A0, 3E0, or 3E1. The dump will include the default storage ranges in those address spaces that are physically-swapped in at the time of the dump. In addition, all storage in ASID 1 and the JES2 address spaces will be dumped. Stand-alone dump will also dump the data spaces created by the DUMPSRV address space.

```
//SADMPGEN JOB MSGLEVEL=(1,1)
//OSG      EXEC PGM=AMDSAOSG
//SYSLIB   DD   SYS1.MACLIB,DISP=SHR
//         DD   SYS1.MODGEN,DISP=SHR
//GENPARMS DD   *
SADXMP3  AMDSADMP CONSOLE=((041,3277),(042,3277),(0A0,3277),           X
               (3E0,3277),(3E1,3277)),                                 X
               DUMP='SP(ALL) IN ASID(1,'JES2') ALSO DATASPACES         X
               OF ASID('DUMPSRV')',                                    X
               IPL=D222,                                               X
               MINASID=PHYSIN,                                         X
               OUTPUT=D450,                                            X
               REUSEDS=NEVER,                                          X
               PROMPT,                                                 X
               VOLSER=SADMPM
         END
/*
```

The output from AMDSAOSG contains a listing for the stand-alone dump common communication table (CCT) and device and dump options (DDO) control blocks that contain information specified at generation time. The remainder of the output consists of messages, including message AMD064I, from both stand-alone dump and, when the residence volume is direct access, the device utility ICKDSF. AMDSAOSG returns the following codes in message AMD064I:

Table 4-2. AMDSAOSG Return Codes

Return Code	Explanation
0	Residence volume initialized
4	Residence volume not initialized due to an error, or a warning was issued during AMDSADMP assembly
8	Residence volume not initialized; GENPRINT could not be opened

Reference

See *z/OS MVS System Messages, Vol 1 (ABA-AOM)* for more information about AMD064I.

Considerations when Using One-Step Generation

When generating the stand-alone dump program using one-step generation, do the following:

- Ensure that the SYSLIB DDNAME concatenates SYS1.MODGEN to SYS1.MACLIB. Your installation should catalog the SYS1.MODGEN data set before generating the stand-alone dump program. Otherwise, the JCL that stand-alone dump produces will fail to create the stand-alone dump program.
- If you are generating stand-alone dump for residence on a direct access volume, AMDSAOSG creates and loads a SYS1.PAGEDUMP.Vvolser data set containing the stand-alone dump program and places an IPL text on the volume. If the volume already contains a SYS1.PAGEDUMP.Vvolser data set, AMDSAOSG will fail. While AMDSAOSG is running, the mount attribute of the volume must be PRIVATE.
- When generating the stand-alone dump program from a Magnetic Tape Subsystem, be aware of which tape format you use or you may not be able to IPL the program. Specifically, IPL processing will end abnormally if you:
  - Generate stand-alone dump on a 3490E Magnetic Tape Subsystem and use a tape subsystem other than a 3490E for IPL.
  - Generate stand-alone dump on a tape subsystem other than a 3490E and use a 3490E Magnetic Tape Subsystem for the IPL.

Two-Stage Generation

In stage-two generation of the stand-alone dump program, you must perform two tasks:

1. Assemble the AMDSADMP macro
2. Initialize the residence volume

Assembling the AMDSADMP Macro

Once you have coded the AMDSADMP macro, you can assemble the macro.

Example: Stage-Two JCL to Assemble the AMDSADMP Macro

Use this JCL to assemble the AMDSADMP macro. The SYSLIB data set must contain the AMDSADMP macro.

```
//ASSEMSAD     JOB       MSGLEVEL=(1,1)
//ASM          EXEC      PGM=ASMA90,REGION=4096K,PARM='DECK'
//SYSLIB       DD        DSN=SYS1.MACLIB,DISP=SHR
//             DD        DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1       DD        UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSPRINT     DD        SYSOUT=(*,,STD),HOLD=YES
//SYSPUNCH     DD        DSN=D10.SYS420.STAGE3.JCL(SADMPST2),DISP=SHR
//SYSLIN       DD        SYSOUT=H
//SYSIN        DD        *
               AMDSADMP   MINASID=ALL,IPL=DSYSDA,                         X
                 DUMP=('DATASPACES OF ASID('XCFAS','CTTX','APPC')'),   X
                 VOLSER=XXXXXX,                                         X
                 CONSOLE=((020,3277),(030,3277),(040,3277),(050,3277)),X
                 PROMPT,MSG=ALL,                                        X
                 OUTPUT=T560
               END
/*
```

The output of the assembly is a job stream that can be used to initialize the residence volume. The output of the assembly can be directed to a DASD or tape device by coding the SYSPUNCH DD card.

To direct the output of the assembly to tape, use the following SYSPUNCH DD statement:

```
//SYSPUNCH DD UNIT=tape,LABEL=(,NL),DISP=(NEW,KEEP),
// VOL=SER=SCRTCH
```

To direct the output of the assembly to a new direct access data set, use the following SYSPUNCH DD statement:

```
//SYSPUNCH DD UNIT=dasd,SPACE=(80,(30,10)),DSN=dsname,
// DISP=(NEW,KEEP),VOL=SER=volser
```

Assembling Multiple Versions of AMDSADMP

You can assemble multiple versions of AMDSADMP at the same time, provided that each version specifies a different residence volume. Differentiate between versions by coding a unique symbol at thebeginning of each macro. AMDSADMP uses the symbol you indicate to create unique stage-two job names. The output from a multiple assembly is a single listing and a single object deck, which can be broken into separate jobs if desired.

Example: Assembling Multiple Versions of AMDSADMP Macro

Use this JCL for coding multiple versions of AMDSADMP.

```
//MULTISAD JOB  MSGLEVEL=(1,1)
//ASM      EXEC PGM=ASMA90,PARM='DECK,NOOBJ'
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR
//         DD   DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD   UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSPRINT DD   SYSOUT=A
//SYSPUNCH DD   SYSOUT=B
//SYSIN    DD   *
TAPE        AMDSADMP IPL=T3400,VOLSER=SADMP1
DASD1       AMDSADMP VOLSER=SADMP2,MINASID=PHYSIN
DASD2       AMDSADMP VOLSER=SADMP3
            END
/*
```

Initializing the Residence Volume

When you are generating stand-alone dump for residence on a direct access volume using the stage-two JCL, AMDSAOSG creates and loads a SYS1.PAGEDUMP.Vvolser data set containing the stand-alone dump program and places an IPL text on the volume.If the volume already contains a SYS1.PAGEDUMP.Vvolser data set, the stage-two job will fail. While the stage-two job is running, the mount attribute of the volume must be PRIVATE.

Physical output from the assembly part of the initialization step is a listing for the stand-alone dump common communication table (CCT) and devices and dump options (DDO) control blocks that contain information specified at generation time. The remainder of the output consists of informational, error, and action messages from both stand-alone dump and, when the residence volume is direct access, the device utility ICKDSF.

When generating the stand-alone dump program from a Magnetic Tape Subsystem, be aware of which tape format you use or you may not be able to IPL the program. Specifically, IPL processing will end abnormally if you:

- Generate stand-alone dump on a 3490E Magnetic Tape Subsystem and use a tape subsystem other than a 3490E for the IPL.
- Generate stand-alone dump on a tape subsystem other than a 3490E and use a 3490E Magnetic Tape Subsystem for the IPL.

Using Two-Stage Generation when Migrating

When migrating to a new version of MVS, IBM recommends that you generate a new version of the stand-alone dump program. Use the new MVS system data sets to build the new version of the stand-alone dump program.

Although the current version of the stand-alone dump program might be able to dump a new version of MVS successfully, it is not guaranteed. MVS may have changed such that the stand-alone dump program would not be able to locate vital information it needs to operate.

To generate a new version of the stand-alone dump program, follow the same steps you followed for a normal two-stage generation, then add the following steps:

- Ensure that the new version of the AMDSADMP macro is being used by specifying the correct SYSLIB data set.
- Use the NUCLIB, MODLIB, LNKLIB and/or ALIB parameters on the AMDSADMP macro invocation to create the correct stage-two JCL.

Example: Stage-Two JCL to Assemble the AMDSADMP Macro

The following output assembles the version of the AMDSADMP macro contained in the SYSLIB data set SYS1.MACLIB, found on a 3390 DASD with volser=NEWSYS. Because the ALIB parameter is specified, the stage-two JCL will use the SYS1.NUCLEUS, SYS1.MODGEN, and SYS1.LINKLIB system data sets, also found on the 3390 DASD with volser=NEWSYS.

```
//ASSEMSAD JOB  MSGLEVEL=(1,1)
//ASM      EXEC PGM=ASMA90,REGION=4096K,PARM='DECK'
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR,
//             UNIT=3390,VOL=SER=NEWSYS
//SYSUT1   DD   UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSPRINT DD   SYSOUT=(*,,STD),HOLD=YES
//SYSPUNCH DD   DSN=D10.SYS430.STAGE3.JCL(SADMPST2),DISP=SHR
//SYSLIN   DD   SYSOUT=H
//SYSIN    DD   *
           AMDSADMP   MINASID=ALL,IPL=DSYSDA,                          X
               DUMP=('DATASPACES OF ASID('XCFAS','CTTX','APPC')'),     X
               VOLSER=SADUMP,                                          X
               CONSOLE=((020,3277),(030,3277),(040,3277),(050,3277)),  X
               PROMPT,MSG=ALL,                                         X
               OUTPUT=T560,                                            X
               ALIB=(NEWSYS,3390)
           END
/*
```

Note: Using the ALIB parameter is convenient if all of the system data sets used by the stand-alone dump program reside on the same volume. Also, note that the same results could have been achieved by coding the NUCLIB, MODLIB, and LNKLIB keywords separately with each specifying NEWSYS and 3390 for volser and unit.

Example: Stage-Two JCL to Assemble the AMDSADMP Macro

The following output assembles the version of the AMDSADMP macro contained in the SYSLIB data set, SYS1.MACLIB, found on a 3390 DASD with volser=NEWSYS. Because the MODLIB parameter is specified, the stage-two JCL will use the SYS1.MODGEN system data set found on a 3380 DASD with volser=SYS51A. Because the ALIB parameter is specified, the stage-two JCL will use the SYS1.NUCLEUS and SYS1.LINKLIB system data sets found on a 3390 DASD with volser=SYS51B.

```
//ASSEMSAD JOB  MSGLEVEL=(1,1)
//ASM      EXEC PGM=ASMA90,REGION=4096K,PARM='DECK'
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR,
//             UNIT=3390,VOL=SER=NEWSYS
//SYSUT1   DD   UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSPRINT DD   SYSOUT=(*,,STD),HOLD=YES
//SYSPUNCH DD   DSN=D10.SYS430.STAGE3.JCL(SADMPST2),DISP=SHR
//SYSLIN   DD   SYSOUT=H
//SYSIN    DD   *
           AMDSADMP   MINASID=ALL,IPL=DSYSDA,                          X
               DUMP=('DATASPACES OF ASID('XCFAS','CTTX','APPC')'),     X
               VOLSER=SADUMP,                                          X
               CONSOLE=((020,3277),(030,3277),(040,3277),(050,3277)),  X
               PROMPT,MSG=ALL,                                         X
               OUTPUT=T560,                                            X
               MODLIB=(SYS51A,3380),                                   X
               ALIB=(SYS51B,3390)
           END
/*
```

Note that the ALIB parameter has no effect on the SYS1.MODGEN system data set because the MODLIB parameter was specified separately.

The stand-alone dump program will be generated using the cataloged system data sets if the NUCLIB, MODLIB, LNKLIB, or ALIB parameters are not specified.

Using Two-Stage Generation for Overriding

When overriding to use a different systems database, IBM recommends that you generate a new version of the stand-alone dump program. Use the new MVS system data sets to build the new version of the stand-alone dump program.

Although the current version of the stand-alone dump program might be able to dump a new version of MVS successfully, it is not guaranteed. MVS may have changed such that the stand-alone dump program would not be able to locate vital information it needs to operate.

To generate a new version of the stand-alone dump program, follow the same steps you followed for a normal two-stage generation, then add the following steps:

- Ensure that the new version of the AMDSADMP macro is being used by specifying the correct SYSLIB data set.
- Use the NUCLIB, MODLIB, LNKLIB and/or ALIB parameters on the AMDSADMP macro invocation to create the correct stage-two JCL.

Example: Stage-Two JCL to Assemble the AMDSADMP Macro with Overrides

The following output assembles the version of the AMDSADMP macro contained in the SYSLIB data set SYS1.MACLIB, found on a 3390 DASD with volser=OVRIDE. Because the ALIB parameter is specified, the stage-two JCL will use the SYS1.NUCLEUS, SYS1.MODGEN, and SYS1.LINKLIB system data sets, also found on the 3390 DASD with volser=OVRIDE.

```
// EXEC ASMAC,PARM.C='DECK,NOOBJECT'
//C.SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR,UNIT=3390,VOL=SER=OVRIDE
//C.SYSPUNCH DD DSN=SMITH.TEST.CNTL(SADMP#2),DISP=OLD
//C.SYSIN DD *
        AMDSADMP IPL=D3390,                  IPL FROM DASD                   X
               VOLSER=SADIPL,                VOL=SER=SADIPL                  X
               OUTPUT=(D330,SYS1,SADMP),     DEFAULT OUTPUT DEVICE           X
               MSG=ALL,                      ALL MESSAGES TO CONSOLE         X
               DUMP=('SP(ALL) IN ASID(1) ALSO DATASPACES OF ASID(1,'JESXCF',
               'APPC','SMSVSAM',  'CONSOLES','SYSBMAS') ALSO PAGETABLES OF
               DATASPACES                                                    X
               ALSO SP(ALL) IN ASID('JES2')'),                               X
               MINASID=ALL,                  INCLUDE SWAPPED OUT SPACES      X
               REUSEDS=CHOICE,               PROMPT FOR DATASET REUSE        X
               DDSPROMPT=NO,                 OVERRIDE BUILD DATASETS         X
               ALIB=(OVRIDE,3390)
           END
/*
```

Note: Using the ALIB parameter is convenient if all of the system data sets used by the stand-alone dump program reside on the same volume. Also, note that the same results could have been achieved by coding the NUCLIB, MODLIB, and LNKLIB keywords separately with each specifying NEWSYS and 3390 for volser and unit.

Running the Stand-Alone Dump Program

The operator usually takes a stand-alone dump for one of the following types of problems:
- Disabled wait
- Enabled wait
- Loop
- Partial system hang

When one of these problems occurs, the stand-alone dump program, residing in the SYS1.PAGEDUMP.Vvolser data set, can be run to produce a stand-alone dump. There are several procedures that can be used to run the stand-alone dump program:
- “Procedure A: Initialize and Run Stand-Alone Dump” on page 4-37
- “Procedure B: Restart Stand-Alone Dump” on page 4-40
- “Procedure C: ReIPL Stand-Alone Dump” on page 4-41
- “Procedure D: Dump the Stand-Alone Dump Program” on page 4-41

When to Use What Procedure:
- Use procedure A to initialize the stand-alone dump program and dump storage.
- If you want to run stand-alone dump again, for instance when stand-alone dump fails, use procedure B, procedure C, or procedure D.
- When you want to restart stand-alone dump, try procedure B before you try procedure C or D.

- Procedures C and D can result in the loss of some central storage from the output, whereas procedure B usually does not.

Although the stand-alone dump program was created under the operating system, it runs as a stand-alone operation.

Procedure A: Initialize and Run Stand-Alone Dump

1. Stop all processors. **Do not** clear storage.
2. Select a processor that was online when the system was stopped.
3. If the processor provides a function to IPL a stand-alone dumpwithout performing a manual STORE STATUS, use this function to IPL stand-alone dump. If you do not use such a function, perform a STORE STATUS before IPLing stand-alone dump. If the operator does not store status, virtual storage is not dumped.

   The hardware store status facility stores the current program status word (PSW), current registers, the processor timer, and the clock comparator into the unprefixed prefix save area (PSA). This PSA is the one used before the nucleus initialization program (NIP) initialized the prefix register.

   If you IPL the stand-alone dump program from the hardware console, it is not necessary to perform the STORE STATUSoperation. Status is automatically stored when stand-alone dump is invoked from the hardware console and automatic store status is on.

   If a STORE STATUS is not done before IPLing a stand-alone dump, the message “ONLY GENERAL PURPOSE REGS VALID” might appear on the formatted dump. The PSW, control registers, etc., are not included in the dump.

   Note

   Do **not** use the LOAD CLEAR option. Using the LOAD CLEAR option erases main storage, which means that you will not be able to diagnose the failure properly.

4. Ready the residence device. If it is a tape, mount the volume on a device attached to the selected processor and ensure that the file-protect ring is in place. If it is a DASD volume, ensure that it is write-enabled.
5. IPL stand-alone dump.

   Stand-alone dump does not communicate with the operator console. Instead, stand-alone dump loads an enabled wait PSW with wait reason code X'3E0000'. The IPLing of the stand-alone dump program causes absolute storage (X'0'-X'18' and storage beginning at X'FC0') to be overlaid with CCWs. You should be aware of this and not consider it as a low storage overlay.

   Note: Stand-alone dump uses the PSW to communicate with the operator or system programmer.

   Stand-alone dump waits for a console I/O interrupt or an external interrupt.
6. When stand-alone dump is IPLed, you may specify a load parm that alters the operation of stand-alone dump. The format of the load parm is S*adddd*.

   The constant S must be specified as the first character or the load parm will be ignored.

The *a* specification allows stand-alone dump to start using a console without the operator performing any action on it. It also allows stand-alone dump to bypass the prompts for which output device and default dump title to use. You may specify the following values for *a*:

N No console communication requested. Use default dump device and title. Execution begins with no console messages. No prompting to the operator is allowed. If a prompt occurs, a wait state will be loaded.

O Use the default console with the default dump device and title. No prompting to the operator is allowed. If a prompt occurs, a wait state will be loaded.

M Use the default console with the default dump device and title. Additional prompts may be made to the operator if they are needed.

C Use the default console. The operator must respond to all prompts.

P Wait for an interrupt from the console device that is to be used. If you do not supply the load parm, this is the default.

The *dddd* specification is the default console device. It must be one of the devices specified as a console device on the AMDSADMP macro when the stand–alone dump was generated, or the constant SYSC for the hardware system console. If you do not specify a default console device, then the stand–alone dump will use the first console defined on the AMDSADMP macro when the stand–alone dump was generated.

The AMDSADMP macro allows you to specify SYSC as the first console in the console list. If you do this without specifying a console device in the load parm, the hardware system console will be the default console device.

If you do not use the load parm, select the system console or an operator console with a device address that is in the console list that you specified at stand-alone dump generation time (in the CONSOLE keyword of AMDSADMP). At stand-alone dump run time, the operator can choose either a console specified with the CONSOLE= keyword or the system console to control stand-alone dump operation. If an operator console is chosen, press ATTENTION or ENTER on that console. (On some consoles, you might have to press RESET first.) This causes an interruption that informs stand-alone dump of the console's address. Message AMD001A appears on the console.

a. Ready an output device. When you dump to devices that have both real and virtual addresses (for example, dumping a VM system), specify only the real address to the stand-alone dump program. If you are dumping to tape, ensure that the tape cartridge is write-enabled. If you are dumping to DASD, ensure that the DASD data set has been initialized using the AMDSADDD REXX utility.
b. Reply with the device number for the output device. If you are dumping to a DASD device and DDSPROMPT=YES was specified on the AMDSADMP macro, message AMD002A is issued to prompt the operator for a dump data set. IF DDSPROMPT=NO was specified, message AMD002A is not issued and the stand-alone dump program assumes that the dump data set name is SYS1.SADMP.

   Notes:

   1) Pressing ENTER in response to message AMD001A will cause the stand-alone dump program to use the default device specified on the OUTPUT= keyword of the AMDSADMP macro. If the default device is a

DASD device, then pressing the ENTER key in response to message AMD001A will cause the stand-alone dump program to use both the default device and the dump data set specified on the OUTPUT= keyword of the AMDSADMP macro. If no dump data set name was provided on the OUTPUT= keyword and the DDSPROMPT=YES keyword was specified, message AMD002A is issued to prompt the operator for a dump data set. If DDSPROMPT=NO was specified, then the stand-alone dump program assumes that the dump data set name is SYS1.SADMP.

2) If you reply with the device number of an attached device that is not of the required device type, or if the device causes certain types of I/O errors, stand-alone dump might load a disabled wait PSW. When this occurs, useprocedure B to restart stand-alone dump.

c. Stand-alone dump prompts you, with message AMD011A, for a dump title.

Example: Using a Load Parm to Perform a Stand-Alone Dump

In this example, the dump is initialized using a load parm with no console prompts.

```
AMD083I   AMDSADMP:  STAND-ALONE DUMP INITIALIZED
AMD101I   OUTPUT DEVICE: 0330 SADMP1 SYS1.SADMP
          SENSE ID DATA: FF 3990 E9 3390 0A  BLOCKSIZE:  24,960
AMD005I   DUMPING OF REAL STORAGE NOW IN PROGRESS.
AMD005I   DUMPING OF REAL STORAGE COMPLETED (MINIMAL).
AMD005I   DUMPING OF REAL STORAGE COMPLETED (SUMMARY).
AMD005I   DUMPING OF REAL STORAGE COMPLETED (IN-USE).
AMD005I   DUMPING OF REAL STORAGE COMPLETED.
AMD108I   DUMPING OF SUMMARY    ADDRESS SPACES COMPLETED.
AMD108I   DUMPING OF SWAPPED IN ADDRESS SPACES COMPLETED.
AMD056I   DUMPING OF VIRTUAL STORAGE COMPLETED.
AMD104I         DEVICE VOLUME USED DATA SET NAME
          1      0330   SADMP1 43% SYS1.SADMP _
```

7. When no console is available, run stand-alone dump without a console.
   a. Ready the default output device that was specified on the OUTPUT parameter on the AMDSADMP macro. For tapes, ensure that the tape cartridge is write-enabled. For DASD, ensure that the dump data set has been initialized using the AMDSADDD REXX utility.
   b. Enter an external interruption on the processor that stand-alone dump was IPLed from. Stand-alone dump proceeds using the default output device and/or the default dump data set. No messages appear on any consoles; stand-alone dump uses PSW wait reason codes to communicate to the operator.
8. When stand-alone dump begins and finishes dumping central storage, it issues message AMD005I to display the status of the dump. stand-alone dump may end at this step.
9. When stand-alone dump begins dumping real storage it issues message AMD005I. Message AMD095I is issued every 30 seconds to illustrate the process of the dump. Message AMD005I will be issued as specific portions of real storage have been dumped, as well as upon completion of the real dump. Stand-alone dump may end at this step.
10. If you specified PROMPT on the AMDSADMP macro, stand-alone dump prompts you for additional storage that you want dumped by issuing message AMD059D.

11. Stand-alone dump dumps instruction trace data, paged-out virtual storage, the stand-alone dump message log, and issues message AMD095I every 30 seconds to illustrate the progress of the dump.
12. When stand-alone dump completes processing, stand-alone dump unloads the tape, if there is one, and enters a wait reason code X'410000'.

Reference

See *z/OS MVS System Codes* for more information about the wait state reason codes loaded into the PSW.

Note: Some S/390® Enterprise Server models do not allow selection of a specific processor to IPL from. Normally, the processor previously IPL'ed will be selected again for this IPL.

Procedure B: Restart Stand-Alone Dump

A system restart does not always work, either because it occurs at a point when stand-alone dump internal resources are not serialized, or because stand-alone dump has been too heavily damaged to function. If the restart does not work, try procedure C (reIPL).

If a dump to a DASD data set is truncated because there is not enough space on the data set to hold the dump, use a system restart to dump the original data to tape. By causing a system restart, you can reinitialize and restart a failing stand-alone dump program without losing the original data you wanted to dump.

Once the output is obtained, the maximum number of times that you can restart the stand-alone dump program is five.

If a permanent error occurs on the output device, the stand-alone dump program will prompt the operator to determine if a restart of the stand-alone dump program should be performed. If the operator indicates that a restart of the stand-alone dump program should be performed, then the stand-alone dump program will restart the dump using the same console and will prompt the operator to specify a different output device. Continue procedure A at step 6A; see 4-38.

For other types of stand-alone dump errors and wait states, it may be necessary for the operator to perform a manual restart of the stand-alone dump program. In this case, the operator should perform the following steps:

1. Perform a system restart on the processor that you IPLed stand-alone dump from.
2. If the restart is successful, stand-alone dump dumps central storage. If stand-alone dump abnormally ends while dumping central storage, try to restart stand-alone dump. If the restart succeeds, stand-alone dump reruns the entire dump. It will first enter wait state X'3E0000' to allow you to specify a new console and output device. You can do this to recover from an I/O error on the output device. Stand-alone dump recognizes any console in the console list and starts with the same output device defaults that are used at the IPL of stand-alone dump.
3. Continue procedure A at step 5, see 5 on page 4-37.

Note: Some S/390 Enterprise Server models do not allow selection of a specific processor to IPL from. Normally, the processor previously IPL'ed will be selected again for this IPL.

Procedure C: ReIPL Stand-Alone Dump

When you reIPL stand-alone dump, the previous running of stand-alone dump has already overlaid some parts of central storage and modified the page frame table. If the virtual storage dump program was in control, a reIPL might not dump paged-out virtual storage. The number of times that you can IPL stand-alone dump to dump paged-out virtual storage is equal to the number of processors present.

To run procedure C, repeat procedure A, but **do not** issue a STORE STATUS.When you are IPLing using a stand-alone dump hardware function, the STORE STATUS is omitted from all IPLs of stand-alone dump after the first IPL. If the previous IPL of stand-alone dump did not load a wait state and reason code of X'250000' orhigher and the reIPL succeeds, stand-alone dump usually completes processing as in procedure A. Some storage locations might not reflect the original contents of central storage because, during a previous IPL, stand-alone dump overlaid the contents. These locations include the absolute PSA and possibly other PSAs.

Procedure D: Dump the Stand-Alone Dump Program

Use a new IPL of stand-alone dump to debug stand-alone dump if stand-alone dump fails. When you use stand-alone dump to dump itself, the dump program dumps central storage only, because a dump of central storage provides enough information to diagnose a stand-alone dump error. Follow procedure A at step 3 by performing a STORE STATUS instruction. Stand-alone dump follows procedure A through step 8, then issues message AMD088D. This message allows the operator to stop the dump after central storage has been dumped or to continue dumping virtual storage.

The self-dumps and system restart are two features of stand-alone dump error recovery. When errors occur during running of a virtual storage dump, stand-alone dump can take a maximum of five self-dumps. You can use these to diagnose stand-alone dump processing errors. The stand-alone dump system restart capability also helps you to test and debug stand-alone dump.

A Stand-Alone Self-Dump

When running a virtual storage dump and stand-alone dump error recovery detects errors in stand-alone dump, stand-alone dump may take a self-dump before proceeding. At most, stand-alone dump takes five self-dumps; on the sixth request for a self-dump, stand-alone dump processing ends. Stand-alone dump places both the self-dump and the operating system dump onto the output tape or DASD.

You can use the LIST subcommand of IPCS to print stand-alone dump self-dumps. The format of the subcommand is:

`LIST` *address* `COMPDATA(AMDSA00x)`

where x = 1 - 5.

Reference

See *z/OS MVS IPCS Commands* for more information.

Running the Stand-Alone Dump Program in a Sysplex

The operator usually takes a stand-alone dump in a sysplex when an MVS system is not responding. Situations that indicate that stand-alone dump should be run include:

- Consoles do not respond

- MVS is in a WAIT state
- An MVS system is in a “status update missing” condition and has been or is waiting to be removed from the sysplex
- A stand-alone dump has been requested by Level 2.

There are two high-level methods for taking a stand-alone dump of an MVS system that resides in a sysplex. Both methods emphasize the expeditious removal of the failing MVS system from the sysplex. If the failed MVS system is not partitioned out of the sysplex promptly, some processing on the surviving MVS systems might be delayed.

Method A

Use this method to take a stand-alone dump of an MVS system that resides in a sysplex. Assume that the MVS system to be dumped is “SYSA”.

1. Perform the STOP function to place the SYSA CPUs into the stopped state.
2. IPL the stand-alone dump program on SYSA (see “Running the stand-alone dump Program”).
3. Issue VARY XCF,SYSA,OFFLINE from another active MVS system in the sysplex if message IXC402D or IXC102A is not already present.

   You do not have to wait for the stand-alone dump to complete before issuing the VARY XCF,SYSA,OFFLINE command.
4. Reply DOWN to message IXC402D or IXC102A.

Performing steps 3 and 4 immediately after IPLing stand-alone dump will expedite sysplex recovery actions for SYSA and allow resources held by SYSA to be cleaned up quickly, thus enabling other systems in the sysplex to continue processing.

Once stand-alone dump is IPLed, MVS cannot automatically ISOLATE system SYSA via SFM, so message IXC402D or IXC102A will be issued after the VARY XCF,SYSA,OFFLINE command or after the XCF failure detection interval expires. You must reply DOWN to IXC402D or IXC102A before sysplex partitioning can complete.

Note: DO NOT perform a SYSTEM RESET in response to IXC402D, IXC102A after the IPL of stand-alone dump. The SYSTEM RESET is not needed in this case because the IPL of stand-alone dump causes a SYSTEM RESET. Once the IPL of stand-alone dump is complete, it is safe to reply DOWN to IXC402D or IXC102A.

Method B

If there is a time delay between performing the STOP function and IPLing the stand-alone dump program (steps 1 and 2 in Method A) use this method. Using this method will expedite the release of resources held by system SYSA while you are preparing to IPL stand-alone dump.

1. Perform the STOP function to place the SYSA CPUs into the stopped state.
2. Perform the SYSTEM RESET-NORMAL function on SYSA
3. Issue VARY XCF,SYSA,OFFLINE from another active MVS system in the sysplex if message IXC402D or IXC102A is not already present.
4. Reply DOWN to message IXC402D, IXC102A

   Performing steps 3 and 4 immediately after doing the SYSTEM RESET will expedite sysplex recovery actions for SYSA and allow resources held by SYSA to be cleaned up quickly, thus enabling other systems in the sysplex to continue processing.

5. IPL the stand-alone dump program. (See “Running the stand-alone dump Program”.) This step can take place at any time after step 2.

Once a SYSTEM RESET is performed, MVS cannot automatically ISOLATE system SYSA via SFM, so message IXC402D or IXC102A will be issued after the VARY XCF,SYSA,OFFLINE command or after the XCF failure detection interval expires. You must reply DOWN to IXC402D or IXC102A before sysplex partitioning can complete.

Note: DO NOT IPL stand-alone dump more than once. This action will invalidate the dump of MVS. To restart stand-alone dump processing, perform the CPU RESTART function on the CPU where the stand-alone dump program was IPLed.

Capturing a Stand-Alone Dump Quickly

There are times when you need to process stand-alone dump information quickly to diagnose a problem. It is important to perform the stand-alone dump process quickly, to minimize the time the system is unavailable. Sometimes a stand-alone dump may not be captured due to the time that the dumping process takes. Skipping the stand-alone dump, however, can prevent the diagnosis of the system failure. Instead of skipping the stand-alone dump, it is better to spend a short time to get as much of the stand-alone dump as is possible, as quickly as possible. The following are two methods to save time when performing a stand-alone dump.

- Minimize the operator actions
- Get a partial stand-alone dump

Minimize the Operator Actions

Time spent waiting for the operator to reply to a message or mount a tape is idle time. Minimizing the operator actions turns the idle time into data capture time. It also simplifies the process, so that the stand-alone dump process becomes easier to do. The following are ways to minimize the operator’s actions when performing a dump.

- Use the stand-alone dump LOAD parameter "SO" or "SM" to skip the prompt for the console to use to avoid other responses to messages.
- Use the default device specified on the OUTPUT= keyword of the AMDSADMP macro. If the default device is a DASD device, then pressing the ENTER key in response to message AMD001A will cause the stand-alone dump program to use both the default device and the dump data set specified on the OUTPUT= keyword of the AMDSADMP macro.
- Use REUSEDS=ALWAYS on the AMDSADMP macro to indicate that stand-alone dump should reuse the dump data set on the specified output device when it determines that the data set is valid, however, it may contain data from a previous dump. Or, you can always clear the dataset.

  Note: Be sure you do not overwrite another dump.
- Specify DDSPROMPT=NO, then the stand-alone dump program assumes that the dump data set name is SYS1.SADMP.
- Do not specify PROMPT on the AMDSADMP macro, unless requested by IBM.
- Use ″fast″ device for output

Get a Partial Stand-Alone Dump

While it is always best to get a complete stand-along dump, sometimes time constraints will not allow this. There is no guarantee that it will be possible to diagnose a failure from a partial stand-alone dump; however, if the choice is between no dump at all or a partial dump, then the partial dump is the best choice.

When taking a partial stand-alone dump:

- Let the stand-alone dump run for as long as you can. If you run out of time, you can stop the dump cleanly.
- Stand-alone dump tries to write out the most important information first.
  - Status information (PSW, registers, and so forth) for all CPUs
  - Critical real storage, including common storage and trace information
  - Real storage for address spaces executing at the time of the dump
  - Any remaining real storage
  - Paged out storage for swapped in address spaces
  - Paged out storage for swapped out address spaces
- Use the EXTERNAL INTERRUPT key to terminate the dumping process. This causes a clean stop, closing the output dataset properly.

Example: Terminating a Stand-Alone Dump

In this example, the dump was ended early using the EXTERNAL INTERRUPT key..

```
AMD083I   AMDSADMP:  STAND-ALONE DUMP RESTARTED
AMD094I   0330 SADMP1 SYS.SADMP
          IS VALID, HOWEVER, IT MAY ALREADY CONTAIN DATA FROM A PREVIOUS DUMP.
          THE INSTALLATION CHOSE TO ALWAYS REUSE THE DUMP DATA SET.
AMD101I   OUTPUT DEVICE: 0330 SADMP1 SYS1.SADMP
          SENSE ID DATA: FF 3990 E9 3390 0A  BLOCKSIZE: 24,960
AMD005I   DUMPING OF REAL STORAGE NOW IN PROGRESS.
AMD005I   DUMPING OF REAL STORAGE COMPLETED (MINIMAL).
AMD005I   DUMPING OF REAL STORAGE COMPLETED (SUMMARY).
AMD089I   DUMP TERMINATED DUE TO EXTERNAL KEY
AMD066I   AMDSADMP ERROR, CODE=0012, PSW=040810008101235E, COMPDATA9AMDSA002)
```

Copying, Viewing, and Printing Stand-Alone Dump Output

When stand-alone dump processing completes the dump, the output resides on either a tape volume, a DASD, or a combination of devices. The easiest way to view the dump is to copy the dump to a DASD data set. When a stand-alone dump resides on multiple devices and/or dump data sets, you can concatenate the dump into one data set. Once the dump is available on DASD, it can be viewed online using IPCS.

Note: If the dump resides in a DASD dump data set, IBM recommends that you copy the dump to another data set for IPCS processing and clear (reinitialize) the dump data set using the AMDSADDD utility. See “Using the AMDSADDD Utility” on page 4-23 for more information.

Copying the Dump to a Data Set

If you want to view the dump online, copy the dump to a data set. There are two tools you can use to copy the dump:

- Use the IPCS COPYDUMP subcommand when the IPCS environment has been set up on your system.
- Use the IEBGENER utility when the IPCS environment has not been set up on your system. Many operators take a stand-alone dump so that the system programmer can view the dump. The operator does not require IPCS on the system because the operator will not be viewing the dump. Therefore, the operator should use the IEBGENER utility to copy the dump to a data set accessible by the system programmer's system.

References

- See *z/OS MVS IPCS Commands* for information about COPYDUMP.
- See *z/OS DFSMSdfp Utilities* for information about IEBGENER.

Copying from Tape

The example below shows how to use IEBGENER to copy tape output to DASD. Two advantages of copying stand-alone dump tape output to DASD are:

- When stand-alone dump ends prematurely and does not give the stand-alone dump output (SYSUT1) an end-of-file, the SYSUT2 data set does contain an end-of file.(SYSUT2 is the data set to which stand-alone dump output is copied.) This occurs even when SYSUT2 is another tape. IEBGENER might end with an I/O error on SYSUT1; this is normal if SYSUT1 does not contain an end-of-file.
- Making SYSUT2 a direct access data set to use as input to IPCS saves IPCS processing time.

Example: Copying Stand-Alone Dump Output from Tape to DASD

Use the following JCL to invoke the IEBGENER utility, which will copy the stand-alone dump output from tape to a DASD data set.

```
//SADCOPY  JOB  MSGLEVEL=(1,1)
//COPY     EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT1   DD   DSN=SADUMP.TAPE,UNIT=tape,
//         VOL=SER=SADOUT,LABEL=(,NL),DISP=SHR,
//         DCB=(RECFM=FBS,LRECL=4160,BLKSIZE=29120)
//SYSUT2   DD   DSN=SADUMP.COPY,UNIT=dasd,
//         VOL=SER=SADCPY,DISP=(NEW,CATLG),
//         DCB=(RECFM=FBS,LRECL=4160,BLKSIZE=24960),
//         SPACE=(4160,(8000,4000),RLSE)
```

Copying from DASD

The example below shows how to use IEBGENER to copy DASD output to a DASD data set. Once the dump is successfully copied, use the AMDSADDD REXX utility to clear (reinitialize) the dump data set and ready it for another stand-alone dump. See “Using the AMDSADDD Utility” on page 4-23 for more information.

Example: Copying Stand-Alone Dump Output from DASD to DASD

Use the following JCL to invoke IEBGENER, which will copy the stand-alone dump output from a DASD data set to another DASD data set.

```
//SADCOPY  JOB  MSGLEVEL=(1,1)
//COPY     EXEC PGM=IEBGENER
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
//SYSUT1   DD   DSN=SYS1.SADMP,UNIT=DASD,
//         VOL=SER=SADMP1,DISP=SHR
//SYSUT2   DD   DSN=SYS2.SADMP,UNIT=DASD,
//         DISP=(NEW,CATLG),
//         VOL=SER=SADMP2,
//         DCB=(LRECL=4160,BLKSIZE=20800,RECFM=FBS,DSORG=PS),
//         SPACE=(CYL,(90,0),RLSE)
```

Copying from Multiple Dump Data Sets

The stand-alone dump program allows a dump to be contained in multiple dump data sets. Therefore, when you want to view a stand-alone dump using IPCS, it is necessary to concatenate all of the dump data sets onto one DASD data set.

Example: Copying Stand-Alone Dump Output from Multiple DASD Data Sets

Use the following JCL to invoke the IPCS COPYDUMP subcommand to copy stand-alone dump output from three DASD dump data sets to another data set. Note that two of the dump data sets reside on the volume SADMP1, while the third resides on the volume SADMP2.

```
//SADCOPY  JOB MSGLEVEL=(1,1)
//COPY     EXEC PGM=IKJEFT01
//IPCSDDIR DD DISP=SHR,DSN=D10IPCS.IPCS.DMPDIR
//SYSTSPRT DD SYSOUT=A
//COPYFROM DD DSN=SADMP1.DDS1,DISP=SHR,UNIT=DASD,VOL=SER=SADMP1
//         DD DSN=SADMP1.DDS2,DISP=SHR,UNIT=DASD,VOL=SER=SADMP1
//         DD DSN=SYS1.SADMP,DISP=SHR,UNIT=DASD,VOL=SER=SADMP2
//COPYTO   DD DSN=SADUMP.COPY,UNIT=DASD,
//            VOL=SER=SADCPY,DISP=(NEW,CATLG),
//            DCB=(RECFM=FBS,LRECL=4160,BLKSIZE=4160),
//            SPACE=(4160,(8000,4000),RLSE)
//SYSTSIN DD  *
  IPCS NOPARM
  COPYDUMP OUTFILE(COPYTO) INFILE(COPYFROM) NOCONFIRM
  END
/*
```

Example: Copying Stand-Alone Dump Output from DASD and Tape

Use the following JCL to invoke the IPCS COPYDUMP subcommand to copy stand-alone dump output from two DASD dump data sets and two tape volumes to a DASD data set.

```
//SADCOPY  JOB MSGLEVEL=(1,1)
//COPY     EXEC PGM=IKJEFT01
//IPCSDDIR DD DISP=SHR,DSN=D10IPCS.IPCS.DMPDIR
//SYSTSPRT DD SYSOUT=A
//COPYFROM DD  DSN=SYS1.SADMP.MAIN.DDS1,DISP=SHR,UNIT=DASD,
//             VOL=SER=SADMP1
//         DD  DSN=SYS1.SADMP.ALTERNAT.DDS1,DISP=SHR,UNIT=DASD,
//             VOL=SER=SADMP2
//COPYTO   DD  DSN=SADUMP.COPY,UNIT=DASD,
//             VOL=SER=SADCPY,DISP=(NEW,CATLG),
//             DCB=(RECFM=FBS,LRECL=4160,BLKSIZE=20800),
//             SPACE=(4160,(8000,4000),RLSE)
//SYSTSIN DD  *
  IPCS NOPARM
  COPYDUMP OUTFILE(COPYTO) INFILE(COPYFROM) NOCONFIRM
  END
/*
```

Viewing Stand-Alone Dump Output

You can view the stand-alone dump output at a terminal using IPCS. Do the following:

1. Start an IPCS session.
2. On the IPCS Primary Option Menu panel, select the SUBMIT option to copy the dump and do initial dump analysis.
3. Return to the IPCS Primary Option Menu panel. Select the DEFAULTS option.
4. IPCS displays the IPCS Default Values panel. Enter the name of the data set containing the dump on the Source line.

5. Return to the IPCS Primary Option Menu panel. Select the BROWSE, ANALYZE, or COMMAND option to view the dump.

Reference

See *z/OS MVS IPCS Commands* for information about the IPCS subcommands.

Printing Stand-Alone Dump Output

You can print an analysis of the stand-alone dump or the entire dump using IPCS.

To print an analysis of the dump in batch mode:

1. Start an IPCS session.
2. On the IPCS Primary Option Menu panel, select the SUBMIT option to copy the dump and do initial dump analysis.
3. On the IPCS MVS/ESA™ Dump Batch Job Option Menu panel, enter the requested information.
4. On the next panel, enter the sysout output class. IPCS writes the dump analysis to the specified output class.
5. The system prints the dump in the printout of the output class.

To print the full dump in batch mode:

1. Use IPCS CLIST BLSCBSAP.

Reference

See *z/OS MVS IPCS User's Guide* for IPCS panels and the CLIST BLSCBSAP.

Example: Printing an Unformatted Stand-Alone Dump

The following example runs an IPCS CLIST that:

- Copies the stand-alone dump from the tape data set defined in an IEFRDER DD statement to a cataloged, direct access data set named SA1DASD.
- Analyzes and formats the dump.
- Writes the formatted dump output to a data set named IPCSPRNT. A TSO/E CLIST used for IPCS should allocate this print output data set to a sysout print class, as follows:

```
ALLOCATE DDNAME(IPCSPRNT) SYSOUT(A)
```

After the CLIST runs, the dump remains available in the SA1DASD data set for supplementary formatting jobs.

```
//PRINTJOB  JOB  MSGLEVEL=1,REGION=800M
//IPCS      EXEC IPCS,CLIST=BLSCBSAP,DUMP=SA1DASD
//IEFPROC.IEFRDER  DD DSN=SA1,DISP=OLD,UNIT=3490
//            VOL=SER=12345,LABEL=(1,NL)
/*
```

Message Output

There are three types of message output from a stand-alone dump program, as follows:

- MNOTES from the AMDSADMP macro
- Messages on the 3480, 3490, or 3590 display
- Messages on the system console or the operator console

Reference

For more information about messages on the system console or the operator console, use LookAt or see *MVS System Messages*. For a description of LookAt, see “Using LookAt to look up message explanations” on page xviii.

Stand-Alone Dump Messages on the 3480, 3490, or 3590 Display

When stand-alone dump output is sent to a 3480, 3490, or 3590 magnetic tape subsystem, stand-alone dump uses the subsystem’s eight-character message display to inform and prompt the operator. The leftmost position on the message display indicates a requested operator action. The eighth position (rightmost) gives additional information.

In the messages listed below, alternating indicates that there are two messages which are flashing on the display, one after the other. A blinking message is one message that is repeated on the display.

The stand-alone dump messages that can appear on the display are:

Dvolser (alternating)

MSADMP#U
Informs the operator that a labeled tape has been rejected and a new tape must be mounted.

MSADMP#U (blinking)
Requests that the operator mount a new tape.

RSADMP#U (blinking)
Indicates that the stand-alone dump program has finished writing to the tape.

RSADMP# (alternating)

MSADMP#U
Informs the operator that an end-of-reel condition has occurred and a new tape must be mounted.

SADMP# (blinking)
Indicates that the tape is in use by stand-alone dump.

SADMP# (alternating)

NTRDY
Informs the operator that some type of intervention is required.

The symbols used in the messages are:

A variable indicating the actual number of cartridges mounted for stand-alone dump. It is a decimal digit starting at 1 and increasing by 1 after each end-of-cartridge condition. When the # value exceeds 9, it is reset to 0.

D Demount the tape and retain it for further system use, for example as a scratch tape. Stand-alone dump does not write on the tape.

M Mount a new tape.

R Demount the tape and retain it for future stand-alone dump use.

U The new tape should not be file-protected.

volser A variable indicating the volume serial number on the existing tape label.

Analyzing Stand-Alone Dump Output

This section describes how to analyze the output from a stand-alone dump. A stand-alone dump can indicate the following types of problems:
- Enabled wait state
- Disabled wait state
- Enabled loop
- Disabled loop

Use the information in this section to determine the type of problem the system has encountered. Once the problem type is determined, see *z/OS MVS Diagnosis: Procedures* for further information about diagnosing the problem type.

The topics in this section are:

- "Collecting Initial Data"
- "Analyzing an Enabled Wait" on page 4-53
- "Analyzing a Disabled Wait" on page 4-57
- "Analyzing an Enabled Loop" on page 4-58
- "Analyzing a Disabled Loop" on page 4-58
-
- "Problem Data Saved by First Level Interrupt Handlers" on page 4-60

Collecting Initial Data

When an operator takes a stand-alone dump, it is important to determine the conditions of the system at the time the dump was taken. Because a stand-alone dump can be requested for a various number of problem types, the collection of problem data is imperative to determining the cause of the error.

The objectives for analyzing the output of a stand-alone dump are as follows:
- Gather symptom data
- Determine the state of the system
- Analyze the preceding system activity
- Find the failing module and component

Gathering External Symptoms

When a stand-alone dump is taken, you must first question the operator or the person who requested the dump. It is important to understand the external symptoms leading up to the system problem. What was noticed before stopping the system? The answer might give you an idea of where the problem lies.

Here are a few questions you should find an answer to before continuing:
- Was the system put into a wait state?
- Were the consoles hung or locked up?
- Were commands being accepted on the master console without a reply?
- Was a critical job or address space hung?

Gathering IPCS Symptoms

After getting a list of symptoms, use IPCS to collect further symptom data. A primary symptom string is usually not available in a stand-alone dump; however, IPCS may add a secondary symptom string.

Example: VERBEXIT SYMPTOMS Output

In the following output, the explanation of the secondary symptom string indicates an enabled wait state condition.

```
                        * * * *  S Y M P T O M  * * * *
ASR10001I The dump does not contain a primary symptom string.
  Secondary Symptom String:

    WS/E000 FLDS/ASMIORQR VALU/CPAGBACKUP FLDS/IOSCOD VALU/CLCLC0D45
    FLDS/IOSTSA VALU/CLCLDEV02

    Symptom            Symptom data     Explanation
    ---------------    -------------    -----------
    WS/E000            000              Enabled wait state code
    FLDS/ASMIORQR      ASMIORQR         Data field name
    VALU/CPAGBACKUP    PAGBACKUP        Error related character value
```

Determining the System State

There are several control blocks you can view that describe the state of the system when the stand-alone dump was requested.

Control Block	Explanation
CSD	Describes the number of active central processors and whether the alternate CPU recovery (ACR) is active.
PSA	Describes the current environment of a central processor, its work unit, FRR stack, an indication of any locks held.
LCCA	Contains save areas and flags of interrupt handlers.
CVT	Contains pointers to other system control blocks.

Use the IPCS subcommand STATUS WORKSHEET to obtain the data that will help you determine the state of the system.

Example: STATUS WORKSHEET Output

In the following output, look for the following:
- The CPU bit mask, which indicates how many processors are online.
- The PSW at the time of the dump
- The PSATOLD. If the fields are zero, this indicates that an SRB is running and the address in SMPSW indicates the save area of the dispatcher. If the fields are non-zero, the address in PSWSV indicates the save area of the dispatcher.
- The PSAAOLD, which indicates what address space jobs are running in.

```
                             MVS Diagnostic Worksheet
Dump Title: SYSIEA01 DMPDSENQ 7/20/93
CPU Model 2064 Version 00 Serial no. 145667 Address 00
Date: 03/20/2001   Time: 05:41:26  Local
 SYSTEM RELATED DATA

 CVT  SNAME (154) ESYS       VERID (-18)
      CUCB   (64) 00FD4B68   PVTP  (164) 00FE4A10    GDA  (230) 01BE1168
      RTMCT (23C) 00F81198   ASMVT (2C0) 00FD8030    RCEP (490) 012AA3F0

 CSD  Available CPU mask: C000  Alive CPU mask: C000  No. of active CPUs: 0002

PROCESSOR RELATED DATA

 NAME             OFFSET  |  CPU 00    CPU 01
--------------------------+----------------------------------------
 PSW at time of dump      | 070E0000  070C9000
                          | 00000000  8124EE9C
 CR0 Interrupt mask       | 5EB1EE40  5EB1EE40
 CR6 I/O class mask       | FE        FE
------ LCCA --------------+----------------------------------------
 IHR1   Recursion    208  | 00        00
 SPN1/2 Spin         20C  | 0000      0000
 CPUS   CPU WSAVT    218  | 00F4BA00  00F6F550
 DSF1/2 Dispatcher   21C  | 0000      0080
 CRFL   ACR/LK flgs  2B4  | 00000000  00000000
------ PSA ---------------+----------------------------------------
 TOLD   Curr TCB     21C  | 00000000  00000000
 ANEW   ASCB         220  | 00FD3BC0  00F56180
 AOLD   Curr ASCB    224  | 00FD3280  00F56180
 SUPER  Super Bits   228  | 04000000  00000000
 CLHT   Lock Table   280  | 00FD4890  00FD4890
 LOCAL  Local lock   2EC  | 00000000  00F0D700
 CLHS   Locks held   2F8  | 00000000  00000001
 CSTK   FRR stack    380  | 00F4D4D0  00000C00
 SMPSW  SRB Disp PSW 420  | 070C0000  070C0000
                     424  | 81142B60  82039000
 PSWSV  PSW Save     468  | 070E0000  070E0000
                     46C  | 00000000  00000000
 MODE   Indicators   49F  | 08        04
```

You can also obtain the stored status of each central processor using the IPCS subcommand STATUS CPU REGISTERS. Watch for these bits in the first half of the PSW:
- Bits 6 and 7 indicate a disabled (04xxxxxx) or enabled (07xxxxxx) condition
- Bit 14 could indicate a wait (000A0000)
- Bits 16 and 17 indicate primary, secondary, access register (AR) or Home mode

Example: STATUS CPU REGISTERS Output

In the following output, the PSW indicates an enabled wait state condition. The program is running in primary mode with 24-bit addressing (bits 16 and 17 are 00 and the second word begins with 0).

```
CPU(X'00') STATUS:
 PSW=070E0000  00000000  NO WORK WAIT
  ASCB1 at FD3280, JOB(*MASTER*), for the home ASID
  ASXB1 at FD34F8 for the home ASID. No block is dispatched
 CLTE: 01CB00E8
  +0000  BLSD..... 00000000  XDS...... 00000000  XRES..... 00000000
  +000C  XQ....... 00FD4900  ESET..... 00FD4908  ULUT..... 00FD4910
  CURRENT FRR STACK IS: SVC
  PREVIOUS FRR STACK(S): NORMAL

  GPR VALUES
      0-3  00000000  00000000  00000000  00000000
      4-7  00000000  00000000  00000000  00000000
      8-11 00000000  00000000  00000000  00000000
     12-15 00000000  00000000  00000000  00000000

  ACCESS REGISTER VALUES
      0-3  006FB01F  00000000  00000000  00000000
      4-7  00000000  00000000  00000000  00000000
      8-11 00000000  00000000  00000000  00000000
     12-15 00000000  00000000  806FA03C  00000000
```

To obtain other fields from important control blocks, use the IPCS subcommand CBFORMAT.

Reference

See *z/OS MVS IPCS Commands* for information about the CBFORMAT subcommand.

You can also use the WHERE subcommand to identify particular areas in the dump. For example, if a general purpose register contains an address, use the WHERE subcommand to determine in what module that address resides.

Example: WHERE Subcommand Output

In the following output, the WHERE subcommand indicates that the address is part of the READONLY nucleus.

```
NOCPU ASID(X'0001') 0124EE9C. IEANUC01.IGVSLIS1+0ADC IN READ ONLY NUCLEUS
```

Analyzing an Enabled Wait

An enabled wait is also known as a dummy wait or a no work wait. An indication of an enabled wait is a PSW of **070E0000 00000000** or **07060000 00000000 00000000 00000000** and GPRs containing all zeroes. An enabled wait occurred when the dispatcher did not find any work to be dispatched. An enabled wait can occur because of resource contention or system non-dispatchability, among other errors.

Reviewing Outstanding I/O Requests

When analyzing a stand-alone dump for an enabled wait condition, check the status of the input/output requests. A display of the IOS control block and any active UCBs can help determine what was happening when the system entered the wait state.

Example: IOSCHECK ACTVUCBS Subcommand Output

In the following output, the HOTIO field indicates that a solicited interrupt has completed with other than DCC-3 since the last time HOT-I/O detection was called. Note also that the IOQF and IOQL fields are identical, indicating that the first and last request for this device is the same.

```
                      * * * ACTVUCBS   Processing * * *
UCB AT 00F8B798:  DEVICE  001; SUBCHANNEL 0001
 UCBPRFIX: 00F8B768
    -0030  RSTEM.... 00        RSV...... 08        MIHTI.... 40
    -002D  HOTIO.... 40        IOQF..... 00F7BC00  IOQL..... 00F7BC00
    -0024  SIDA..... 0001      SCHNO.... 0001      PMCW1.... 2888
    -001E  MBI...... 0000      LPM...... 80        RSV...... 00
    -001A  LPUM..... 80        PIM...... 80        CHPID.... 21000000
    -0014            00000000  LEVEL.... 01        IOSF1.... 00
    -000E  MIHCT.... 0000      LVMSK.... 00000001  LOCK..... 00000000
    -0004  IOQ...... 00F7BC00
```

Analyzing for Resource Contention

You can obtain information related to resource contention by using the IPCS subcommand ANALYZE. This subcommand displays contention information for I/O, ENQs, suspend locks, allocatable devices and real storage.

Example: ANALYZE Subcommand Output

In the following output, 61 units of work are waiting to be processed. The top RB is in a wait state.

```
                 CONTENTION EXCEPTION REPORT
JOBNAME=*MASTER*  ASID=0001  TCB=006E8E88
JOBNAME=*MASTER* HOLDS THE FOLLOWING RESOURCE(S):

   RESOURCE #0011:There are 0061 units of work waiting for this resource
      NAME=MAJOR=SYSIEA01 MINOR=DMPDSENQ SCOPE=SYSTEM

 STATUS FOR THIS UNIT OF WORK:
   This address space is on the SRM IN queue.
   Task non-dispatchability flags from TCBFLGS4:
    Top RB is in a wait
```

Obtaining Real Storage Data

Use the IPCS RSMDATA subcommand to obtain information about storage usage and any unusual condition that may have occurred prior to requesting the stand-alone dump. In the RSMDATA output, if the percent usage field is 100%, there are no frames left. Also, the percent of available total fixed frames should not be a high number. If it is, there may be a program using too many resources to complete.

Example: RSMDATA Output

In the following output, the percent of available total fixed frames is at 25%.

```
                     R S M   S U M M A R Y   R E P O R T
                          Tot real Below Prf real Dbl real      Expanded
                          -------- ----- -------- -------- -------------
In configuration . . . .    33,792 4,096   33,742        -        49,152
Available for allocation    32,672 4,089   33,742      120        49,152
Allocated  . . . . . . .    32,398 3,964   33,483      113        48,594
Percent usage  . . . . .        99    96       99       94            98
Common fixed frames  . .     3,087   317    3,087        -             -
  Percent of available .         9     7        9        -             -
Total fixed frames . . .     8,338   907        -        -             -
  Percent of available .        25    22        -        -             -
```

You can also check the ASM control blocks to determine the statistics applicable to I/O requests. The I/O requests received and completed should be the same.

Example: ASMCHECK Output

In the following output, note that the 509577 I/O requests received have all been completed.

```
ASMVT AT 00FD8030
509577 I/O REQUESTS RECEIVED, 509577 I/O REQUESTS COMPLETED BY ASM
240487 NON-SWAP WRITE I/O REQUESTS RECEIVED, 240487 NON-SWAP WRITE I/O
       REQUESTS COMPLETE
PART AT 01CB5310
PAGE DATA SET 0 IS ON UNIT 15B
PAGE DATA SET 1 IS ON UNIT 15B
PAGE DATA SET 3 IS ON UNIT 14A
PAGE DATA SET 4 IS ON UNIT 150
PAGE DATA SET 5 IS ON UNIT 15B
```

Determining Dispatchability

By performing an address space analysis on the major system address space, you can determine if there is any work waiting and if the address space is dispatchable. The major address space you should analyze are:

- Master scheduler, ASID 1
- Console
- JES2/JES3
- IMS/CICS/VTAM

When you are analyzing an address space for dispatchability, keep in mind these questions:

- Are there any suspended SRBs on the queue?

  You will need to run the WEBs on ASCBSAWQ and look for WEBs that have a WEBFLAG1 field

  of X'000000' to check if there are any SRBs ready to be dispatched.
- Are there any ready TCBs indicated by ASCBTCBS and ASCBTCBL?

  ASCBTCBS and ASCBTCBL contain a count of the number of TCBs containing ready work to be dispatched. To find the TCBs for ASCBTCBL, look at the WEBs on the ASCBLTCS and ASCBLTCB queues that belong to the home space.

- If there is ready work, is the ASCB dispatchable (ASCBDSP1)?

  ASCBDSP1 is a non-dispatchability flag. See *z/OS MVS Data Areas, Vol 1 (ABEP-DALT)* for more information about what the values of ASCBDSP1 indicate.
- If there is no ready work, are the TCBs in a normal wait (TCBFLGS4, TCBFLGS5, TCBNDSP)?

  A non-zero value in any of these fields indicates that the TCB is non-dispatchable.

Example: SUMMARY FORMAT Output

In the following output, ASCBDSP1 is X'04', indicating that this address space is not eligible for CML lock requests. The ASCBSAWQ, ASCBLTCN, and ASCBTCBS fields all contain zeroes, indicating that there is no ready work available.

```
ASCB: 00FD2B80
   +0000  ASCB..... ASCB      FWDP..... 00FC4400  BWDP..... 00000000
   +000C  LTCS..... 00000000  SVRB..... 00F4FBA8  SYNC..... 000727F4
   +0018  IOSP..... 00000000  WQID..... 0000      SAWQ..... 00000000
   +0024  ASID..... 0001      LL5...... 00        HLHI..... 01
   +002A  DPH...... 01FF      LDA...... 7F748EB0  RSMF..... C0
   +0038  CSCB..... 00000000  TSB...... 00000000
   +0040  EJST..... 0000009F  94659288
   +0048  EWST..... AEE06377  45A41803            JSTL..... 000141DE
   +0054  ECB...... 00000000  UBET..... 00000000  TLCH..... 00000000
   +0060  DUMP..... 00699D90  AFFN..... FFFF      RCTF..... 01
   +0067  FLG1..... 00        TMCH..... 00000000  ASXB..... 00FD2EA8
   +0070  SWCT..... 47BE      DSP1..... 00        FLG2..... CE
   +0076  SRBS..... 0000      LLWQ..... 00000000  RCTP..... 00000000
   +0080  LOCK..... 00000000  LSWQ..... 00000000  QECB..... 00000000
   +008C  MECB..... 00000000  OUCB..... 015178E8  OUXB..... 01517BF0
   +0098  FMCT..... 0000      LEVL..... 03        FL2A..... 00
   +009C  XMPQ..... 00000000  IQEA..... 00000000  RTMC..... 00000000
   +00A8  MCC...... 00000000  JBNI..... 00000000  JBNS..... 00FD2B18
   +00B4  SRQ1..... 00        SRQ2..... 00        SRQ3..... 00
   +00B7  SRQ4..... 00        VGTT..... 00CD7458  PCTT..... 1AB6F008
   +00C0  SSRB..... 0000      SMCT..... 00        SRBM..... 07
   +00C4  SWTL..... 00000000  SRBT..... 000015D1  E5E32000
   +00D0  LTCB..... 00000000  LTCN..... 00000000  TCBS..... 00000000
   +00DC  LSQT..... 00000000  WPRB..... 00FD2E90  NDP...... FF
   +00E5  TNDP..... FF        NTSG..... FF        IODP..... FF
   +00E8  LOCI..... 00000000  CMLW..... 00000000  CMLC..... 00000000
   +00F4  SSO1..... 000000    SSO4..... 00        ASTE..... 02900040
   +00FC  LTOV..... 7FFD8400  ATOV..... 7FFDCCA8  ETC...... 0007
   +0106  ETCN..... 0000      LXR...... 0007      AXR...... 0000
   +010C  STKH..... 00FD35C0  GQEL..... 00000000  LQEL..... 00000000
   +0118  GSYN..... 00000000  XTCB..... 006A3D90  CS1...... 00
   +0121  CS2...... 00        GXL...... 02449430
   +0128  EATT..... 0000000E  DAC0D475
   +0130  INTS..... AED8EC7B  0C7C0900            LL1...... 00
   +0139  LL2...... 00        LL3...... 00        LL4...... 00
   +013C  RCMS..... 00000000  IOSC..... 0000450A  PKML..... 0000
   +0146  XCNT..... 01F4      NSQA..... 00000000  ASM...... 00FD3520
   +0150  ASSB..... 00FD2D00  TCME..... 00000000  GQIR..... 00000000
   +0168  CREQ..... 00000000  RSME..... 02219120  AVM1..... 00
   +0171  AVM2..... 00        AGEN..... 0000      ARC...... 00000000
   +0178  RSMA..... 02219000  DCTI..... 0066E2EE
```

Reference

See *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)* for the mapping structure of WEBs under the IHAWEB.

If your address space analysis indicated that ready work was available to be dispatched, look at ASCBDSP1 to determine if the address space is dispatchable. If your address space analysis indicated that there was no ready work available to be dispatched, look at the TCBs to determine if they are in a normal wait.

Example: SUMMARY FORMAT Output

In the following output, the TCB fields indicate that the top RB is in a wait.

```
TCB: 00FD3608
   +0000  RBP...... 006FF048  PIE...... 00000000  DEB...... 00000000
   +000C  TIO...... 00000000  CMP...... 00000000  TRN...... 00000000
   +0018  MSS...... 7F7463A0  PKF...... 00        FLGS..... 00008004  00
   +0022  LMP...... FF        DSP...... FF        LLS...... 006FFD38
   +0028  JLB...... 00000000  JPQ...... 006FF200
 GENERAL PURPOSE REGISTER VALUES
     0-3  00000001  000027C4  00009FBC  00000004
     4-7  006FFF48  006FEFB8  00F6E900  0000005C
     8-11 80001E52  00C0DCE8  006F5FF0  00FCF778
    12-15 00FCF170  006FF348  80FCF1C0  806FF048
   +0070  FSA...... 00000000  TCB...... 006FF6F0  TME...... 00000000
   +007C  JSTCB.... 00FD3608  NTC...... 00000000  OTC...... 00000000
   +0088  LTC...... 006FF6F0  IQE...... 00000000  ECB...... 00000000
   +0094  TSFLG.... 00        STPCT.... 00        TSLP..... 00
   +0097  TSDP..... 00        RD....... 7F748F04  AE....... 7F746280
   +00A0  STAB..... 00F0B860  TCT...... 00000000  USER..... 00000000
   +00AC  NDSP..... 00000000  MDIDS.... 00000000  JSCB..... 00C0BE84
        .
        .
        .
   +014C  BDT...... 00000000  NDAXP.... 00000000  SENV..... 00000000
   Task non-dispatchability flags from TCBFLGS4:
    Top RB is in a wait
```

Analyzing a Disabled Wait

A disabled wait condition can be analyzed by checking the PSW at the time of the error. If bits 6 and 7 are zero and bit 14 contains a 1, there is a disabled wait. The wait state code is in byte 7, with the reason code in byte 5.

Example: Determining the Wait State Code

In the following PSW, the wait state code is X'014' and the reason code is zero.

```
PSW=000E0000  00000014
```

In another example, the wait state code is X'064' and the reason code is X'09'.

```
PSW=000A0000  00090064
```

In z/Architecture mode, the PSW would look like:

```
PSW=0002000 00000000 00000000 00090064
```

Stand-Alone Dump

Once you determine the wait state code from the PSW, look at the documentation for the specific wait state code for any action you can take.

Reference

See *z/OS MVS System Codes* for the specific wait state code you encountered.

If you cannot find the wait state code documented, do one of the following:

- Analyze the dump to determine if it is a stand-alone dump wait state.
- Check PSASMPSW and PSAPSWSV to determine if the dispatcher loaded the wait state PSW because of an overlay. See Chapter 7, "The Dump Grab Bag" on page 7-1 for more information about storage overlays.
- Use the stored status registers to determine who loaded the wait state into the PSW.

Analyzing an Enabled Loop

To determine if the stand-alone dump was requested because of an enabled loop, you need to view the system trace table. Repetitive patterns in the system trace table indicate an enabled loop condition. An enabled loop, however, does not normally cause a system outage. It will cause an outage in these circumstances:

- There is a non-preemptible loop in SRB mode
- There is a loop in a high priority address space that is in TCB mode

Example: SYSTRACE Output

In the following output, the CLKC entries indicate an enabled loop, and because column three is all zeroes, this loop is in SRB mode. The PSW addresses on the CLKCs identify the looping program. Use the WHERE subcommand to locate the responsible program.

```
01 003E 00000000   CLKC      070C0000 8100765C  00001004 00000000
01 003E 00000000   CLKC      070C2000 81005638  00001004 00000000
01 003E 00000000   CLKC      070C0000 810056E6  00001004 00000000

01 003E 00000000   CLKC      070C0000 80FF0768  00001004 00000000

01 003E 00000000   CLKC      070C0000 80FE4E34  00001004 00000000

01 003E 00000000   CLKC      070C1000 81004BB8  00001004 00000000
```

Because of interrupt processing that occurs during an enabled loop, the stored status data might not point to the module causing the loop. To determine if a first level interrupt handler (FLIH) was active, view the PSASUPER field of the PSA. If the PSASUPER field is non-zero, a FLIH was active at the time of the error. Using the FLIH's save area, find the PSW and registers at the time of the error. The address in the second half of the PSW will point to the module involved in the loop. See "Problem Data Saved by First Level Interrupt Handlers" on page 4-60 for more information.

Analyzing a Disabled Loop

A disabled loop is not visible in the system trace output because disabled routines do not take interrupts. Normally, a disabled loop results in a spin loop in a multiprocessor environment. When analyzing a stand-alone dump for a disabled loop, use the stored status data to determine the module involved in the loop. Also, examine the in-storage logrec buffer for entries that recovery routines have made but which were not written to the logrec data set because of a system problem. Very often it is these records that are the key to the problem solution. See “Obtaining Information from the Logrec Recording Control Buffer” on page 14-18 for more information.

If a disabled loop is suspected, have the requestor initiate a CPUTRACE before taking the stand-alone dump. CPUTRACE formats loop trace records through the instruction address trace on the SCP control frame.

Example: CPUTRACE Output

In the following output:

POSITION is the order of the instruction in the trace. IPCS discards any address that is 2, 4, or 6 greater than the previous address; use the position number to determine how many instructions are omitted.

ADDRESS is the address of the instruction.

MODULE is the name of the module containing the instruction plus the offset of the instruction into the module.

```
               * * * * CONSOLE INITIATED LOOP RECORDING * * * *
CPU(01) Recording
 DATE/TIME   09/18/86 05:00:51

 PSW=070E0000  00000000  NO WORK WAIT

  GPR VALUES
      0-3  00000000  00000000  00000000  00000000
      4-7  00000000  00000000  00000000  00000000
      8-11 00000000  00000000  00000000  00000000
     12-15 00000000  00000000  00000000  00000000

  CONTROL REGISTER VALUES
      0-3  5EB1EE40  03F7E07F  02401080  00000001
      4-7  00010001  03C8D040  FE000000  03F7E07F
      8-11 00000000  00000000  00000000  00000000
     12-15 01E0AAF8  03F7E07F  DF883C88  01A14008

 INSTRUCTION TRACE

  POSITION ADDRESS  MODULE
  -------- -------- --------------------------------------------
  00000001 01095F00 IEANUC01.IEAVEEXT+18
  0000000A 01095F22 IEANUC01.IEAVEEXT+3A
  00000027 01096592 IEANUC01.IEAVEEXT+06AA
  .
  .
  .
 MODULES FOUND IN TRACE RECORDS

  CPU(01)

  IEANUC01.IEAVEEXT
  IEANUC01.IEAVETRC
  IEANUC01.AHLMCIH
  IEANUC01.IEAVEXS
```

SLIP Problem Data in the SLIP Work Area

In a stand-alone dump taken after a SLIP ACTION=WAIT trap matches, problem data can be found in a work area pointed to by the PSAWTCOD field in the prefix save area (PSA).

Offset	Length	Content
0(0)	1	RTM/SLIP processing environment indicator: X'01': RTM1 X'02': RTM2 X'03': MEMTERM X'04': PER
1(1)	2	Logical processor identifier (CPUID)
3(3)	1	System mask, if offset 0 is 2 (RTM2)
4(4)	4	Pointer to general purpose registers 0 through 15 at the time of the event
8(8)	4	Pointer to the program status word (PSW) at the time of the event
12(C)	4	One of the following, as indicated by the RTM/SLIP processing environment indicator at offset 0 of the work area: • Pointer to the system diagnostic work area (SDWA), if offset 0 is 1 (RTM1) • Pointer to the recovery termination manager 2 (RTM2) work area (RTM2WA), if offset 0 is 2 (RTM2) • Pointer to the address space control block (ASCB) being ended, if offset 0 is 3 (MEMTERM) • Pointer to the PER code, if offset 0 is 4 (PER)
16(10)	4	Pointer to cross memory information (control registers 3 and 4) at the time of the event
20(14)	4	Pointer to access registers AR0 through AR15 at the time of the event. Pointer to the high 32 bits of the 64-bit GPRs, or 0 if not available. See Wait State 01B in the *z/OS MVS System Codes* for more information.

Problem Data Saved by First Level Interrupt Handlers

If processing is stopped or an error occurs in one of the first level interrupt handlers (FLIH), you might need to determine the PSW and registers of the interrupted program. Field PSASUPER has bits to indicate if one of the FLIH's was in control:
- PSAIO for the IO FLIH
- PSASVC for the SVC FLIH
- PSAEXT for the external FLIH
- PSAPI for the program interrupt FLIH

The following charts show where each of the FLIH's will save PSW and registers for interrupted tasks or SRB's:
-
- "Problem Data Saved for a Program Check for Task and SRB Code" on page 4-61
- "Problem Data Saved by the I/O FLIH for Task and SRB Code" on page 4-62
- "Problem Data Saved by the External FLIH for Task and SRB Code" on page 4-63

Problem Data Saved by the SVC FLIH for Task and SRB Code

Code giving up control	Data saved	Field receiving data	Control block
All SVCs, initially	General purpose registers 7-9	PSAGPREG	PSA
	General purpose registers, if a problem occurred	LCCASGPR	LCCA
All SVCs	PSW	RBOPSW	Requestor's RB
	Cross memory status	XSBXMCRS	XSB
	PCLINK stack header	XSBSTKE	XSB
	EAX	XSBEAX	XSB
	Access registers 0-15	STCBARS	STCB
	Current linkage stack entry pointer	STCBLSDP	STCB
Type 1 and 6 SVCs	General purpose registers 0-15	TCBGRS	TCB
Type 2, 3, and 4 SVCs	General purpose registers 0-15	RBGRSAVE	SVRB

Problem Data Saved for a Program Check for Task and SRB Code

Code giving up control	Data saved	Field receiving data	Control block
Initially for non-recursive program interruptions	General purpose registers 0-15	LCCAPGR2	LCCA
	PSW	LCCAPPSW	LCCA
	ILC/PINT	LCCAPINT	LCCA
	TEA	LCCAPVAD	LCCA
	TEA AR number	LCCAPTR2	LCCA
	Control registers 0-15	LCCAPCR2	LCCA
	Access registers 0-15	LCCAPAR2	LCCA
Initially for recursive program interruptions	General purpose registers 0-15	LCCAPGR1	LCCA
	PSW	LCCAPPS1	LCCA
	ILC/PINT	LCCAPIC1	LCCA
	TEA	LCCAPTE1	LCCA
	TEA AR number	LCCAPTR2	LCCA
	Control registers 0-15	LCCAPCR1	LCCA
	Access registers 0-15	LCCAPAR1	LCCA

Stand-Alone Dump

Code giving up control	Data saved	Field receiving data	Control block
Initially for monitor call interruptions that occur during page fault or segment fault processing	General purpose registers 0-15	LCCAPGR3	LCCA
	PSW	LCCAPPS3	LCCA
	ILC/PINT	LCCAPIC3	LCCA
	TEA	LCCAPTE3	LCCA
	TEA AR number	LCCAPTR3	LCCA
	Control registers 0-15	LCCAPCR3	LCCA
	Access registers 0-15	LCCAPAR3	LCCA
Initially for all trace buffer full interruptions	General purpose registers 0-15	LCCAPGR4	LCCA
For unlocked tasks for page faults or segment faults that require I/O; problem data is moved from the LCCA	Registers		TCB and STCB
	PSW		RB
	Other status		XSB
For locked tasks for page faults or segment faults that require I/O; problem data is moved from the LCCA	Registers		IHSA
	PSW		IHSA
	Other status		XSB for IHSA
For SRBs for page faults or segment faults that require I/O; SRB is suspended, no status is saved			

Problem Data Saved by the I/O FLIH for Task and SRB Code

Code giving up control	Data saved	Field receiving data	Control block
Initially	General purpose registers 0-15	SCFSIGR1	SCFS
	Control registers 0-15	SCFSICR1	SCFS
	Access registers 0-15	SCFSIAR1	SCFS
For unlocked tasks	General purpose registers 0-15	TCBGRS	TCB
	PSW	RBOPSW	RB
	Cross memory status	XSBXMCRS	XSB
	EAX	XSBEAX	XSB
	Access registers 0-15	STCBARS	STCB
	Current linkage stack entry pointer	STCBLSDP	STCB

Code giving up control	Data saved	Field receiving data	Control block
For locally locked tasks	General purpose registers 0-15	IHSAGPRS	IHSA for locked address space
	PSW	IHSACPSW	IHSA for locked address space
	Access registers 0-15	IHSAARS	IHSA for locked address space
	Current linkage stack entry pointer	IHSALSDP	IHSA for locked address space
	Cross memory status	XSBXMCRS	XSB for locked address space
	EAX	XSBEAX	XSB for locked address space
For SRBs and non-preemptive TCBs	General purpose registers 0-15	SCFSIGR1	SCF
	PSW	FLCIOPSW	PSA

Problem Data Saved by the External FLIH for Task and SRB Code

Code giving up control	Data saved	Field receiving data	Control block
Initially	General purpose registers 7-10	PSASLSA	PSA
For locally locked tasks	General purpose registers 0-15	IHSAGPRS	IHSA
	PSW	IHSACPSW	IHSA
	Access registers 0-15	IHSAARS	IHSA
	Current linkage stack entry pointer	IHSALSDP	IHSA
	Cross memory status	XSBXMCRS	XSB
	EAX	XSBEAX	XSB
Unlocked tasks	General purpose registers 0-15	TCBGRS	TCB
	PSW	RBOPSW	RB
	Access registers 0-15	STCBARS	STCB
	Current linkage stack entry pointer	STCBLSDP	STCB
	Cross memory status	XSBXMCRS	XSB
	EAX	XSBEAX	XSB
For SRBs and non-preemptive TCBs	General purpose registers 0-15	SCFSXGR1	SCFS
	PSW	SCFSXPS1	SCFS
	Control registers 0-15	SCFSXCR1	SCFS
	Access registers 0-15	SCFSXAR1	SCFS

Stand-Alone Dump

Code giving up control	Data saved	Field receiving data	Control block
If first recursion	General purpose registers 0-15	SCFSXGR1	SCFS
	PSW	SCFSXPS2	SCFS
	Control registers 0-15	SCFSXCR2	SCFS
	Access registers 0-15	SCFSXAR2	SCFS
If second recursion	General purpose registers 0-15	SCFSXGR3	SCFS
	PSW	FLCEOPSW	PSA
	Control registers 0-15	SCFSXCR3	SCFS
	Access registers 0-15	SCFSXAR3	SCFS

Chapter 5. ABEND Dumps

Just a little picture before the lights go out.

An ABEND dump shows the virtual storage for a program. Typically, a dump is requested when the program cannot continue processing and abnormally ends. An operator can also request an ABEND dump while ending a program or an address space.

The system can produce three types of ABEND dumps, two formatted dumps (SYSABEND and SYSUDUMP) and one unformatted dump (SYSMDUMP), when a program cannot continue processing and a DD statement for an ABEND dump was included in the JCL for the job step that has ended. The data included is dependent on:

- Parameters supplied in the IEAABD00, IEADMR00, and IEADMP00 parmlib members for SYSABENDs, SYSMDUMPs, and SYSUDUMPs, respectively.
- A determination by the system
- ABEND, CALLRTM, or SETRP macro dump options
- The IEAVTABX, IEAVADFM, or IEAVADUS installation exit

A SYSABEND dump can be used for diagnosis of complex errors in any program running under the operating system. A SYSMDUMP can be used for diagnosis of system problems when the dump is requested in a program. A SYSUDUMP can be used for diagnosis of program problems needing simple problem data.

With a SYSABEND, SYSMDUMP, or SYSUDUMP, the system has detected the error and therefore provided a starting point (such as a job step completion code) for analysis.

Major Topics

This chapter covers the following topics, which describe how to use ABEND dumps:

- “Synopsis of ABEND Dumps”
- “Obtaining ABEND Dumps” on page 5-3
- “Printing and Viewing Dumps” on page 5-7
- “Contents of ABEND Dumps” on page 5-8
- “Customizing ABEND Dump Contents” on page 5-14
- “Analyzing an ABEND Dump” on page 5-20

Synopsis of ABEND Dumps

Use the following table as a quick reference for the three types of ABEND dumps. If you need further information about ABEND dumps, refer to the sections following this table.

ABEND Dumps

<table>
<tr><th>Obtaining the Dump</th><th>Receiving the Dump</th><th>Dump Contents</th></tr>
<tr>
<td>SYSABEND:

Assembler macro in any program:
• ABEND with DUMP
• SETRP with DUMP=YES

Assembler macro in an authorized program:
• ABEND with DUMP
• CALLRTM with DUMP=YES
• SETRP with DUMP=YES

Operator command on a console with master authority:
• CANCEL with DUMP

For full information, see “Obtaining ABEND Dumps” on page 5-3.</td>
<td>Formatted dump in a data set with the ddname of SYSABEND:
• In SYSOUT; print in the output class or browse at a terminal
• On tape or direct access: print in a separate job or browse at a terminal
• On a printer (Not recommended; the printer cannot be used for anything else for the duration of the job step.)

For full information, see “Obtaining ABEND Dumps” on page 5-3.</td>
<td>Default contents: summary dump for the failing task and other task data. See “Contents of ABEND Dumps” on page 5-8.

Customized by all of the following:
• IEAADB00 parmlib member
• Parameter list on the requesting ABEND, CALLRTM, or SETRP macro
• Recovery routines invoked by the recovery termination manager (RTM)
• Cumulative from all CHNGDUMP operator commands with SYSABEND
• Installation-written routines at the IEAVTABX, IEAVADFM, and IEAVADUS exits

For full information about customization, see “Customizing ABEND Dump Contents” on page 5-14.</td>
</tr>
<tr>
<td>SYSMDUMP:

Assembler macro in any program:
• ABEND with DUMP
• SETRP with DUMP=YES

Assembler macro in an authorized program:
• ABEND with DUMP
• CALLRTM with DUMP=YES
• SETRP with DUMP=YES

Operator command on a console with master authority:
• CANCEL with DUMP

For full information, see “Obtaining ABEND Dumps” on page 5-3.</td>
<td>Unformatted dump in a data set with the ddname of SYSMDUMP:
• On tape or direct access; use IPCS to format and print/view the dump

For full information, see “Obtaining ABEND Dumps” on page 5-3.</td>
<td>Default contents: summary dump and system data for the failing task. See “Contents of ABEND Dumps” on page 5-8.

Customized by all of the following:
• IEADMR00 parmlib member
• Parameter list on the requesting ABEND, CALLRTM, or SETRP macro
• Recovery routines invoked by the recovery termination manager (RTM)
• Cumulative from all CHNGDUMP operator commands with SYSMDUMP
• Installation-written routines at the IEAVTABX exit

For full information about customization, see “Customizing ABEND Dump Contents” on page 5-14.</td>
</tr>
</table>

Obtaining the Dump	Receiving the Dump	Dump Contents
SYSUDUMP: Assembler macro in any program: • ABEND with DUMP • SETRP with DUMP=YES Assembler macro in an authorized program: • ABEND with DUMP • CALLRTM with DUMP=YES • SETRP with DUMP=YES Operator command on a console with master authority: • CANCEL with DUMP For full information, see “Obtaining ABEND Dumps”.	Formatted dump in a data set with the ddname of SYSUDUMP: • In SYSOUT; print in the output class or browse at a terminal • On tape or direct access; print in a separate job or browse at a terminal • On a printer (Not recommended; the printer cannot be used for anything else for the duration of the job step.) For full information, see “Obtaining ABEND Dumps”.	Default contents: summary dump for the failing task. See “Contents of ABEND Dumps” on page 5-8. Customized by all of the following: • IEADMP00 parmlib member • Parameter list on the requesting ABEND, CALLRTM, or SETRP macro • Recovery routines invoked by the recovery termination manager (RTM) • Cumulative from all CHNGDUMP operator commands with SYSUDUMP • Installation-written routines at the IEAVTABX, IEAVADFM, and IEAVADUS exits For full information about customization, see “Customizing ABEND Dump Contents” on page 5-14.

Obtaining ABEND Dumps

You can obtain SYSABEND, SYSUDUMP, and SYSMDUMP dumps using one process. To obtain a specific type of ABEND dump, specify the correct DD statement in your JCL:

- For SYSABEND dumps:

```
//SYSABEND   DD ...
```

- For SYSUDUMP dumps:

```
//SYSUDUMP   DD ...
```

- For SYSMDUMP dumps:

```
//SYSMDUMP   DD ...
```

Provide a data set to receive the dump, then arrange to view the dump. If a data set is not provided, the system ignores a request for an ABEND dump.

When setting up the data set, determine if it will contain privileged data. If so, protect it with passwords or other security measures to limit access to it.

Data Set for Dump

Define the data set in either:
- The JCL for the job step, for batch processing
- The logon procedure for a TSO/E userid, for foreground processing

Define the data set in a DD statement with a ddname of SYSABEND, SYSMDUMP, or SYSUDUMP. The ddname for the data set determines how the dump can be printed or viewed, what the dump contains, and how the dump contents can be customized. The first two effects are discussed in the following topics.

The system writes the dump in a sequential data set using the basic sequential access method (BSAM). The dump data set can be on any device supported by BSAM. Note that the system provides a data control block (DCB) for the dump data set and opens and closes the DCB.

VIO for ADDRSPC=REAL

A SYSMDUMP DD statement must specify a virtual input/output (VIO) data set if the job or step to be dumped is running in nonpageable virtual storage, that is, the JCL JOB or EXEC statement specifies ADDRSPC=REAL.

Preallocated Data Sets for SYSMDUMP Dumps

You may use any dataset name you wish for the SYSMDUMP dataset. However, the dataset name SYS1.SYSMDPxx will be treated specially. If you use the data set naming convention of SYS1.SYSMDPxx for a DISP=SHR data set, the system writes only the first dump, with all subsequent dump requests receiving system message IEA849I. The data set can be either a magnetic tape unit or a direct access storage device (DASD) data set.

When using this naming convention, you must manage the dump data set to use the same data set repeatedly for SYSMDUMP dumps. For subsequent dumps, you must initialize the SYS1.SYSMDPxx data set with an end-of-file (EOF) mark as the first record.

Naming Convention

You must use SYS1.SYSMDPxx, where xx is 00 through FF and identifies the exact data set to be used.

Data Set Disposition

If you specify DISP=SHR with the SYS1.SYSMDPxx naming convention, the facility that enables the system to write only the first dump becomes active.

If you specify DISP=SHR without the SYS1.SYSMDPxx naming convention, the system writes a new dump over the old dump when the same data set is the target for multiple dumps. This also happens for multiple dumps within the same job if each step does not specify FREE=CLOSE on the SYSMDUMP DD statement.

For dispositions other than DISP=SHR, the system uses the data set as if it were any other MVS data set. If you specify DISP=MOD, the system writes the dump following the previous dump, so that the data set contains more than one dump. If you specify DISP=OLD, the system writes a new dump over the old dump when the same data set is the target for multiple dumps.

Data Set Management

To minimize the loss of subsequent dumps, your installation exit should follow these steps for the management of SYS1.SYSMDPxx data sets:

1. Intercept system message IEA993I. The system issues this message when it writes the dump to the SYS1.SYSMDPxx data set.
2. Copy the dump onto another data set.
3. Clear the SYS1.SYSMDPxx data set by writing an EOF mark as the first record, making it available for the next SYSMDUMP dump to be written on the data set.

The installation exit routine can be one of the following:

- IEAVMXIT
- The exit routine specified on the USEREXIT parameter in the MPFLSTxx parmlib member

References

- See *z/OS MVS System Messages, Vol 6 (GOS-IEA)* for a description of system messages IEA849I and IEA993I.
- See *z/OS MVS Installation Exits* for information about the installation exit routine.

Process for Obtaining ABEND Dumps

Obtain an ABEND dump by taking the following steps for each job step where you want to code a dump:

1. Code a DD statement in the JCL for every job step where a dump would be needed. The statement can specify one of the following:
   - Direct access
   - SYSOUT
   - Tape
   - Printer (Not recommended; printer cannot be used for anything else for duration of job step.)

   Example: Direct Access SYSMDUMP DD Statement

   The following example places a dump on direct access. In the example, SYSDA is an installation group name for direct access storage devices (DASD).

   Like the tape DD statement, the system deletes or keeps the data set depending on how the step ends.

   ```
   //SYSMDUMP  DD  DSN=DUMP2,UNIT=SYSDA,DISP=(,DELETE,KEEP),
   //            SPACE=(TRK,(30,10))
   ```

   Example: SYSOUT SYSABEND DD Statement

   The following example places the dump in sysout output class A. In the example, output class A is a print class. The system prints a dump written to this class when printing the class.

   ```
   //SYSABEND  DD  SYSOUT=A
   ```

Example: Tape SYSUDUMP DD Statement

The following example places a SYSUDUMP dump on a scratch tape.

In the example, TAPE is an installation group name. DEFER specifies that the operator is to mount the tape only when the data set is opened; thus, the operator will not mount the tape unless it is needed for a dump.

The system deletes the data set if the job step ends normally; in this case, the data set is not needed because no dump was written. The system keeps the data set if the step ends abnormally; the data set contains a dump. A future job step or job can print the dump.

```
//SYSUDUMP  DD  DSN=DUMPDS,UNIT=(TAPE,,DEFER),DISP=(,DELETE,KEEP)
```

2. Place the DD statement in the JCL for the job step that runs the program to be dumped or in the logon procedure for a TSO/E userid.

Example: SYSABEND DD Statement in Logon Procedure

The following example shows a SYSABEND DD statement in the logon procedure for a TSO/E userid. A dump statement must appear in the logon procedure in order to process a dump in the foreground.

The system keeps the data set if the job step ends abnormally.

```
//SYSABEND  DD  DSN=MYID3.DUMPS,DISP=(OLD,,KEEP)
```

3. If you need to diagnose a program that does not contain code for an ABEND dump, code one of the following:
   - ABEND assembler macro with a DUMP parameter in a problem program or an authorized program

     Example: ABEND Macro Dump Request

     The following example shows a macro that ends a program with a user completion code of 1024 and requests a dump:

     ```
              ABEND 1024,DUMP
     ```

   - SETRP assembler macro with a DUMP=YES parameter in the recovery routine for a problem program or an authorized program

     Example: SETRP Macro Dump Request

     The following example shows a macro in an ESTAE recovery routine for a problem program. The address of the system diagnostic work area (SDWA) is in register 1, which is the default location.

     ```
              SETRP DUMP=YES
     ```

   - CALLRTM assembler macro with a DUMP=YES parameter in an authorized program

Example: CALLRTM Macro Dump Request

The following example shows a macro in an authorized program. The macro ends a program and requests a dump. Register 5 contains the address of the task control block (TCB) for the program.

```
CALLRTM TYPE=ABTERM,TCB=(5),DUMP=YES
```

4. If you need to diagnose a program that already contains code for an ABEND dump, and that program is already abending, skip step 5.
5. If you need to diagnose a program that already contains code for an ABEND dump, but the program is not currently abending, ask the operator to enter a CANCEL command with a DUMP parameter on the console with master authority.

Example: Canceling a Job and Requesting a Dump

To cancel a job and request a dump, ask the operator to use either of the following:

```
CANCEL  BADJOB,DUMP

CANCEL  STARTING,A=1234,DUMP
```

Example: Canceling a Userid and Requesting a Dump

To cancel a userid and request a dump, ask the operator to use either of the following:

```
CANCEL  U=MYID3,DUMP

CANCEL  U=*LOGON*,A=5678,DUMP
```

6. The system writes a formatted dump to the data set defined in step 1.

Printing and Viewing Dumps

You can print or view the different types of ABEND dumps as follows:

SYSABEND and SYSUDUMP Dumps

These two dumps are formatted as they are created. They can be:

- In a SYSOUT data set. The system can print the dump when printing the output class. To view at a terminal, use a facility that allows the viewing of JES SPOOL data sets.
- On a tape or direct access data set. Print the dump in a separate job or job step or view the dump at a terminal by browsing the data set containing the dump.

  A convenient way to print the dump is in a later job step that runs only if an earlier job step abnormally ends and, thus, requests a dump. For this, use the JCL EXEC statement COND parameter.
- Sent directly to a printer. Note this is not recommended; the printer cannot be used for anything else while the job step is running, whether a dump is written or not.

Example: Using IEBPTPCH to Print a Dump

The following JCL uses the IEBPTPCH facility to print a formatted dump data set. In this example, a SYSABEND dump is printed. The same JCL can be used for a SYSUDUMP. Because the system formats the dump when creating it, the IEBPTPCH utility program can print the dump.

The dump is in a data set named DUMPDS on tape.

```
⋮
//PRINT     EXEC  PGM=IEBPTPCH
//SYSPRINT  DD    SYSOUT=A
//SYSUT1    DD    DSN=DUMPDS,UNIT=TAPE,DISP=(OLD,DELETE)
//SYSUT2    DD    SYSOUT=A
//SYSIN     DD    *
   PRINT    PREFORM=A,TYPORG=PS
/*
```

SYSMDUMP Dumps

This dump is unformatted when created. The system can write the dump to tape or direct access. Use IPCS to format the dump and then view it at a terminal or print it. SYSMDUMP dumps are especially useful for diagnosing errors because IPCS can produce specific information for specific requests.

Reference

See *z/OS MVS IPCS User's Guide* for more information.

Contents of ABEND Dumps

You can specify the contents of an ABEND dump by specifying parameters on the ddname in the JCL for the program. This topic discusses the IBM-supplied default contents and contents available through customization.

All three ABEND dumps contain a summary dump, although the SYSMDUMP summary dump contains less information than the SYSABEND and SYSUDUMP summary dumps. The SYSUDUMP consists of only the summary dump. The SYSABEND dump also contains task data, while the SYSMDUMP also contains system data. The SYSMDUMP dump is a synchronous SVC dump and contains data similar to the data in an SVC dump.

Hiperspaces

ABEND dumps do not include hiperspaces. To include hiperspace in an ABEND dump, read the data from the hiperspace into address space storage that is being dumped.

Adding Areas

If some needed areas are not included by default, see "Customizing ABEND Dump Contents" on page 5-14 for ways to add the areas.

Determining Current ABEND Dump Options

Use a DISPLAY DUMP operator command to get the dump mode and options in effect for SVC dumps and ABEND SYSABEND, SYSMDUMP, and SYSUDUMP dumps. The system displays the mode and options in message IEE857I.

Example: Determining the Mode and Options

To request the mode and options, enter:

```
DISPLAY DUMP,OPTIONS
```

If the options listed are not the ones desired, use a CHNGDUMP operator command to change them.

References

- See *z/OS MVS System Commands* for the DISPLAY and CHNGDUMP operator commands.
- For a description of these messages, use LookAt or see *MVS System Messages*. For a description of LookAt, see “Using LookAt to look up message explanations” on page xviii.

Default Contents of ABEND Dumps

The contents of the three ABEND dumps are detailed in the following two tables. The table below shows dump contents alphabetically by the parameters that specify the areas in the dumps. To select a dump, decide what areas will be used to diagnose potential errors. Find the areas in the tables. The symbols in columns under the dump indicate how the area can be obtained in that dump. The symbols are:

D IBM-supplied default contents

M Available on the macro that requests the dump

P Available in the parmlib member that controls the dump options

X Available on the CHNGDUMP operator command that changes the options for the dump type

Parameter	Dump Contents	ABEND Dump to SYSUDUMP	ABEND Dump to SYSABEND	ABEND Dump to SYSMDUMP
ALL	All the dump options available in a SYSMDUMP dump, except the NOSYM and ALLNUC options			X
ALLNUC	The DAT-on and DAT-off nucleuses			P X
ALLPA	All link pack areas, as follows: • Job pack area (JPA) • Link pack area (LPA) active for the task being dumped • Related Supervisor Call (SVC) modules	M P X	D M P X	M
ALLPDATA	All the program data areas	P X	P X	

ABEND Dumps

Parameter	Dump Contents	ABEND Dump to SYSUDUMP	ABEND Dump to SYSABEND	ABEND Dump to SYSMDUMP
ALLSDATA	All the system data areas	P X	P X	P
ALLVNUC	The entire virtual control program nucleus, including: • Prefixed save area (PSA) • System queue area (SQA) • Local system queue area (LSQA)	M P X	M P X	M
CB	Control blocks for the task being dumped	M P X	D M P X	M
CSA	Common service area (CSA) (that is, subpools 227, 228, 231, 241)			P X
DM	Data management control blocks for the task being dumped: • Data control block (DCB) • Data extent block (DEB) • Input/output block (IOB)	M P X	D M P X	M
ENQ	Global resource serialization control blocks for the task being dumped: • Global queue control blocks • Local queue control blocks	P X	D P X	
ERR	Recovery termination manager (RTM) control blocks for the task being dumped: • Extended error descriptor (EED) for RTM • Registers from the system diagnostic work area (SDWA) • RTM2 work area (RTM2WA) • Set task asynchronous exit (STAE) control block (SCB)	M P X	D M P X	M
GRSQ	Global resource serialization control blocks for the task being dumped: • Global queue control blocks • Local queue control blocks			P X

Parameter	Dump Contents	ABEND Dump to SYSUDUMP	ABEND Dump to SYSABEND	ABEND Dump to SYSMDUMP
IO	Input/output supervisor (IOS) control blocks for the task being dumped: • Execute channel program debug area (EXCPD) • Unit control block (UCB)	M P X	D M P X	M
JPA	Job pack area (JPA): module names and contents	M P X	M P X	M
LPA	Active link pack area (LPA): module names and contents	M P X	M P X	M P X
LSQA	Local system queue area (LSQA) allocated for the address space (that is, subpools 203 - 205, 213 - 215, 223 - 225, 229, 230, 233 - 235, 249, 253 - 255)	M P X	D M P X	D M P X
NOSYM	No symptom dump (message IEA995I)	P X	P X	P X
NUC	Read/write portion of the control program nucleus (that is, only non-page-protected areas of the DAT-on nucleus), including: • Communication vector table (CVT) • Local system queue area (LSQA) • Prefixed save area (PSA) • System queue area (SQA)	M P X	M P X	D M P X
PCDATA	Program call information for the task	M P X	M P X	M
PSW	Program status word (PSW) when the dump was requested	M P X	D M P X	M P X
Q	Global resource serialization control blocks for the task being dumped: • Global queue control blocks • Local queue control blocks	M	M	M

ABEND Dumps

Parameter	Dump Contents	ABEND Dump to SYSUDUMP	ABEND Dump to SYSABEND	ABEND Dump to SYSMDUMP
REGS	Registers at entry to ABEND, that is, when the dump was requested: • Access registers • Floating-point registers • General registers • Vector registers, vector status register, and vector mask register for a task that uses the Vector Facility	M P X	D M P X	M
RGN	Allocated pages in the private area of each address space being dumped, including subpools 0 - 127, 129 - 132, 203 - 205, 213 - 215, 223 - 225, 229, 230, 236, 237, 244, 249, 251 - 255			D P X
SA or SAH	Save area linkage information, program call linkage information, and backward trace of save areas	M P X	D M P X	M
SPLS	Storage allocated in user subpools 0 - 127, 129 - 132, 244, 251, and 252 for the task being dumped Note that SUBPLST in the macro parameter list for a SYSABEND or SYSUDUMP dump overrides SPLS in the dump options list, but only for the dump being requested.	M P X	D M P X	M
SQA	System queue area (SQA) allocated (that is, subpools 226, 239, 245, 247, 248) The control blocks for the failing task in the SQA include: • Address space control block (ASCB) • Job scheduler address space control block (JSAB)	M P X	M P X	D M P X
SUBTASKS	Storage for the task being dumped and program data for all of its subtasks	M P X	M P X	D M

Parameter	Dump Contents	ABEND Dump to SYSUDUMP	ABEND Dump to SYSABEND	ABEND Dump to SYSMDUMP
SUM	Summary dump, see “Default Contents of Summary Dumps in ABEND Dumps”	D M P X	D M P X	D M P X
SWA	Scheduler work area (SWA) (that is, subpools 236 and 237)	M P X	M P X	D M P X
TRT	System trace and generalized trace facility (GTF) trace, as available	M P X	D M P X	
	System trace, as available			D M P X

Default Contents of Summary Dumps in ABEND Dumps

If only a summary dump is requested, as in a SYSUDUMP dump that is not customized, the summary information is together, because it forms the entire dump. When a summary dump is combined with other dump options, the summary dump information is scattered throughout the dump.

In the following table, an S indicates that a summary dump is available with the dump type.

Summary Dump Contents	ABEND Dump to SYSUDUMP	ABEND Dump to SYSABEND	ABEND Dump to SYSMDUMP
Completion code: The system or user completion code if an ABEND macro requested the dump and, if it exists, the accompanying reason code	S	S	S
Control blocks for the failing task, including: • **ASCB** (address space control block) • **CDE** (contents directory entry) • **LLE** (load list element) • **RB** (request block) • **TCB** (task control block) • **TIOT** (task input/output table) • **XL** (extent list)	S	S	
Control blocks for the recovery termination manager (RTM): • **EED** (extended error descriptor) for RTM • **Registers** from the system diagnostic work area (SDWA) • **RTM2WA** (RTM2 work area) • **SCB** (set task asynchronous exit (STAE) control block)	S	S	S
Dump header, mapped by the AMDDATA macro			S
Dump index	S	S	
Dump title: The job and step being dumped, the time and date of the dump, the dump identifier, and the processor	S	S	S
Load module, if the PSW points to an active load module:			

ABEND Dumps

Summary Dump Contents	ABEND Dump to SYSUDUMP	ABEND Dump to SYSABEND	ABEND Dump to SYSMDUMP
• **Name**	S	S	S
• **Module Contents**	S	S	
• **Offset** into the load module of the failing instruction	S	S	
• **Module pointed to in the last PRB** (program request block)	S	S	
PSW (program status word) at entry to ABEND, that is, when the dump was requested The PSW includes the instruction length code and the interrupt code for the failing instruction.	S	S	S
Registers at entry to ABEND, that is, when the dump was requested	S	S	S
Save areas of register contents	S	S	
Storage: 4 kilobytes before and 4 kilobytes after the addresses in the PSW and the registers The dump shows, by ascending address, only the storage that the user is authorized to access. Duplicate addresses are removed.	S	S	S
System trace table entries for the dumped address space	S	S	
TCB summary: Information from the task control blocks (TCB) in the address space being dumped	S	S	
Virtual storage map: The subpools in the address space being dumped: • Subpool number • Subpool key • The owning or sharing task control block (TCB) • The beginning address and length of each allocated area • The beginning address and length of each free area	S	S	

Customizing ABEND Dump Contents

The ddname of the data set for the ABEND dump determines how the contents can be customized.

The system determines the contents of a particular ABEND dump from the options list the system maintains for the type of dump. The dump options list can be customized, cumulatively, by all the ways shown in the following tables. Thus, for example, a SYSMDUMP ABEND dump written for an ABEND macro can be completely different from the default SYSMDUMP ABEND dump described in this document.

References

- See *z/OS MVS Initialization and Tuning Reference* for parmlib members.
- See *z/OS MVS System Commands* for the CHNGDUMP operator command.
- See *z/OS MVS Programming: Assembler Services Reference ABE-HSP* and *z/OS MVS Programming: Assembler Services Reference IAR-XCT* for the ABEND, SETRP, SNAP, SNAPX, ESTAE, ESTAEX, and ATTACH or ATTACHX with ESTAI macros.
- See *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO* and *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* for the SETRP and CALLRTM macros.
- See *z/OS MVS Installation Exits* for the IEAVTABX, IEAVADFM, and IEAVADUS installation exits.

Recommendations for Customizing ABEND Dumps

How an installation customizes dumps should depend on the usual use of each type of dump. The IBM-supplied dump options for ABEND dumps are designed for the following uses:

- SYSABEND dumps: For diagnosis of complex errors in any program running under the operating system
- SYSMDUMP dumps: For diagnosis of system problems when the dump is requested in a program
- SYSUDUMP dumps: For diagnosis of program problems needing simple problem data

For SYSUDUMP dumps, the IBM-supplied IEADMP00 member specifies the default contents as only a summary dump. An installation should consider using the IEADMP00 member as supplied, because it offers a small dump for simple problems.

Program Areas in Dumps

To request a meaningful dump for a particular program, code an ABEND macro that points to a macro parameter list. Specify in the list the data areas that are needed to diagnose the abnormally ending program but that are not specified in the parmlib member for the dump. Two examples are:

- If the task that is ending has subtasks and they might cause an error, specify PDATA=SUBTASKS in the macro parameter list to dump the subtasks.
- To see only the subpools used by the program, specify the subpool numbers in a SUBPLST option for a SYSABEND dump. The SPLS option, which is a default for SYSABEND dumps, writes all user subpools. Leaving SPLS in the dump options may make the dump bigger than needed. Note that SUBPLST in the macro parameter list overrides SPLS in the current dump options.

Nucleus Areas in Dumps

Dump options control the parts of the nucleus that appear in a dump. A diagnostician seldom needs to analyze all of the nucleus. An installation can eliminate nucleus areas from dumps. If the IBM-supplied defaults are used, an SYSMDUMP ABEND dump contains the read/write DAT-on nucleus.

An installation can obtain one copy of the DAT-off nucleus to use in any problem by entering a DUMP operator command.

ABEND Dumps

The ABEND dump options that control dumping of the nucleus areas are:

Dump Option
: **Nucleus Area**

SDATA=NUC
: Read/write DAT-on nucleus

SDATA=ALLNUC
: All of the DAT-on nucleus: read/write and read-only

Customizing SYSABEND Dump Contents

<table>
<tr><th>SYSABEND Customization</th><th>Effect</th><th>Example</th></tr>
<tr>
<td>Replacing IEAABD00 parmlib member (by using the IEBUPDTE utility).</td>
<td>Change occurs: At system initialization

What changes: IEAABD00 contains the IBM-supplied default dump options. Replacing IEAABD00 changes the dump options for SYSABEND.</td>
<td>To add program call data and the link pack area to all SYSABEND dumps, while retaining the IBM-supplied options, use IEBUPDTE to change the IEAABD00 member to contain:<pre>SDATA=(LSQA,CB,ENQ,TRT,ERR,
DM,IO,SUM,PCDATA)
PDATA=(PSW,REGS,SPLS,ALLPA,
SA,LPA)</pre></td>
</tr>
<tr>
<td>Using a macro parameter list.

The DUMPOPT or DUMPOPX parameter on the ABEND or CALLRTM macro points to the parameter list. The list is usually created by a list-form SNAP or SNAPX macro.</td>
<td>Change occurs: At dump request

What changes: The macro parameter list options are added to the dump options list, but only for the dump being requested.

Note that SUBPLST in the macro parameter list overrides SPLS in the dump options list, but only for the dump being requested.</td>
<td>To add program call data and the link pack area to this SYSABEND dump, code in the program:<pre>   ABEND 76,DUMP,
               DUMPOPT=PARMS
PARMS   SNAP SDATA=PCDATA,
               PDATA=LPA,MF=L</pre></td>
</tr>
<tr>
<td>Recovery routines invoked by the recovery termination manager:
FRRs (function recovery routines) for a system component
ESTAE/ESTAI recovery routines established by an ESTAE or ESTAEX macro or the ESTAI parameter of an ATTACH or ATTACHX macro
ARRs (associated recovery routines)
These routines issue SETRP macros. To customize the dump contents, the DUMPOPT or DUMPOPX parameter on the SETRP macro points to a parameter list. The list is usually created by a list-form SNAP or SNAPX macro.</td>
<td>Change occurs: Just before dumping

What changes: The SETRP macro parameter list options are added to the dump options list, but only for the dump being requested.</td>
<td>To add program call data and the link pack area to this SYSABEND dump, code in the recovery routine:<pre>   SETRP ,DUMP=YES,
               DUMPOPT=PARMS
PARMS   SNAP SDATA=PCDATA,
               PDATA=LPA,MF=L</pre></td>
</tr>
</table>

SYSABEND Customization	Effect	Example

Entering CHNGDUMP operator command with SYSABEND parameter on a console with master authority.	**Change occurs**: Immediately when entered
	
	What changes:
	
	For ADD: CHNGDUMP options are added to the IEAABD00 options, previous CHNGDUMP options, and all macro parameter list options. The options remain added until a CHNGDUMP DEL,SYSABEND operator command is entered.
	
	For OVER: CHNGDUMP options override all other dump options.
	
	For DEL: All CHNGDUMP options are deleted and the dump options in IEAABD00 are used again.
	
	When more than one CHNGDUMP operator command with SYSABEND is entered, the effect is cumulative.
		To add program call data and the link pack area to all SYSABEND dumps until changed by CHNGDUMP DEL,SYSABEND, enter:
		`CHNGDUMP SET,ADD,SYSABEND,`
		`SDATA=PCDATA,PDATA=LPA`
		
		To return to the IEAABD00 options, enter:
		`CHNGDUMP DEL,SYSABEND`

Through IEAVTABX installation exit name list.	**Change occurs**: Just before dumping **What changes**: The routine can add or delete options from the dump options, but only for the current dump.	See *z/OS MVS Installation Exits*.
Through IEAVADFM or IEAVADUS installation exits. IEAVADFM is a list of installation routines to be run. IEAVADUS is one installation routine.	**Change occurs**: During dumping. The routine runs during control block formatting of a dump with the CB option. **What changes**: The routine can add control blocks to the dump.	See *z/OS MVS Installation Exits*.

Customizing SYSMDUMP Dump Contents

SYSMDUMP Customization	Effect	Example

Replacing IEADMR00 parmlib member (by using the IEBUPDTE utility).	**Change occurs**: At system initialization
	
	What changes: IEADMR00 contains the IBM-supplied default dump options. Replacing IEADMR00 changes the dump options for SYSMDUMP.
		To add the link pack area to all SYSMDUMP dumps, while retaining all the IBM-supplied defaults, use IEBUPDTE to change the IEADMR00 member to contain:
		`SDATA=(NUC,SQA,LSQA,SWA,TRT,RGN,`
		`SUM,LPA)`

Using a macro parameter list.
 	
The DUMPOPT or DUMPOPX parameter on the ABEND or CALLRTM macro points to the parameter list. The list is usually created by a list-form SNAP or SNAPX macro.
	Change occurs: At dump request
	
	What changes: The macro parameter list options are added to the dump options list, but only for the dump being requested.
		To add the link pack area to this SYSMDUMP dump, code in the program:
		`         ABEND 76,DUMP,`
		`               DUMPOPT=PARMS`
		`PARMS    SNAP  PDATA=LPA,MF=L`

SYSMDUMP Customization	Effect	Example

Recovery routines invoked by the recovery termination manager:
• FRRs (function recovery routines) for a system component
• ESTAE/ESTAI recovery routines established by an ESTAE or ESTAEX macro or the ESTAI parameter of an ATTACH or ATTACHX macro
• ARRs (associated recovery routines)
 	
These routines issue SETRP macros. To customize the dump contents, the DUMPOPT or DUMPOPX parameter on the SETRP macro points to a parameter list. The list is usually created by a list-form SNAP or SNAPX macro.
	Change occurs: Just before dumping
	
	What changes: The SETRP macro parameter list options are added to the dump options list, but only for the dump being requested.
		To add the link pack area to this SYSMDUMP dump, code in the recovery routine:
		`         SETRP ,DUMP=YES,`
		`               DUMPOPT=PARMS`
		`PARMS    SNAP  PDATA=LPA,MF=L`

Entering CHNGDUMP operator command with SYSMDUMP parameter on a console with master authority.	**Change occurs**: Immediately when entered
	
	What changes:
	
	For ADD: CHNGDUMP options are added to the IEADMR00 options, previous CHNGDUMP options, and macro parameter list options. The options remain added until a CHNGDUMP DEL,SYSMDUMP operator command is entered.
	
	For OVER: CHNGDUMP options override all other dump options.
	
	For DEL: All CHNGDUMP options are deleted and the dump options in IEADMR00 are used again.
	
	When more than one CHNGDUMP operator command with SYSMDUMP is entered, the effect is cumulative.
		To add the link pack area to all SYSMDUMP dumps until changed by CHNGDUMP DEL,SYSMDUMP, enter:
		`CHNGDUMP SET,ADD,SYSMDUMP=(LPA)`
		
		To return to the IEADMR00 options, enter:
		`CHNGDUMP DEL,SYSMDUMP`

Through IEAVTABX installation exit name list.	**Change occurs**: Just before dumping **What changes**: The routine can add or delete options from the dump options, but only for the current dump.	See *z/OS MVS Installation Exits*.

Customizing SYSUDUMP Dump Contents

SYSUDUMP Customization	Effect	Example

Replacing IEADMP00 parmlib member (by using the IEBUPDTE utility).	**Change occurs**: At system initialization
	
	What changes: IEADMP00 contains the IBM-supplied default dump options. Replacing IEADMP00 changes the dump options for SYSUDUMP.
		To add program call data and user subpool storage to all SYSUDUMP dumps, while retaining the summary dump, use IEBUPDTE to change the IEADMP00 member to contain:
		`SDATA=(SUM,PCDATA)`
		`PDATA=SPLS`

Using a macro parameter list.
 	
The DUMPOPT or DUMPOPX parameter on the ABEND or CALLRTM macro points to the parameter list. The list is usually created by a list-form SNAP or SNAPX macro.
	Change occurs: At dump request
	
	What changes: The macro parameter list options are added to the dump options list, but only for the dump being requested.
	
	Note that SUBPLST in the macro parameter list overrides SPLS in the dump options list, but only for the dump being requested.
		To add program call data and user subpool storage to this SYSUDUMP dump, code in the program:
		`         ABEND 76,DUMP,`
		`               DUMPOPT=PARMS`
		`PARMS    SNAP  SDATA=PCDATA,`
		`               PDATA=SPLS,MF=L`

Recovery routines invoked by the recovery termination manager:
• FRRs (function recovery routines) for a system component
• ESTAE/ESTAI recovery routines established by an ESTAE or ESTAEX macro or the ESTAI parameter of an ATTACH or ATTACHX macro
• ARRs (associated recovery routines)
 	
These routines issue SETRP macros. To customize the dump contents, the DUMPOPT or DUMPOPX parameter on the SETRP macro points to a parameter list. The list is usually created by a list-form SNAP or SNAPX macro.
	Change occurs: Just before dumping
	
	What changes: The SETRP macro parameter list options are added to the dump options list, but only for the dump being requested.
		To add program call data and user subpool storage to this SYSUDUMP dump, code in the recovery routine:
		`         SETRP ,DUMP=YES,`
		`               DUMPOPT=PARMS`
		`PARMS    SNAP  SDATA=PCDATA,`
		`               PDATA=SPLS,MF=L`

ABEND Dumps

SYSUDUMP Customization	Effect	Example

Entering CHNGDUMP operator command with SYSUDUMP parameter on a console with master authority.	**Change occurs**: Immediately when entered
	
	What changes:
	
	For ADD: CHNGDUMP options are added to the IEADMP00 options, previous CHNGDUMP options, and all macro parameter list options. The options remain added until a CHNGDUMP DEL,SYSUDUMP operator command is entered.
	
	For OVER: CHNGDUMP options override all other dump options.
	
	For DEL: All CHNGDUMP options are deleted and the dump options in IEADMP00 are used again.
	
	When more than one CHNGDUMP operator command with SYSUDUMP is entered, the effect is cumulative.
		To add program call data and user subpool storage to all SYSUDUMP dumps until changed by CHNGDUMP DEL,SYSUDUMP, enter:
		`CHNGDUMP SET,ADD,SYSUDUMP,`
		`SDATA=PCDATA,PDATA=SPLS`
		
		To return to the IEADMP00 options, enter:
		`CHNGDUMP DEL,SYSUDUMP`

Through IEAVTABX installation exit name list.	**Change occurs**: Just before dumping **What changes**: The routine can add or delete options from the dump options, but only for the current dump.	See *z/OS MVS Installation Exits*.
Through IEAVADFM or IEAVADUS installation exits. IEAVADFM is a list of installation routines to be run and IEAVADUS is one installation routine.	**Change occurs**: During dumping. The routine runs during control block formatting of a dump with the CB option. **What changes**: The routine can add control blocks to the dump.	See *z/OS MVS Installation Exits*.

Analyzing an ABEND Dump

Note: A SYSMDUMP ABEND dump is always a synchronous SVC dump. To analyze a SYSMDUMP, see “Analyzing an SVC Dump” on page 2-36.

ABEND dumps written to SYSABEND and SYSUDUMP data sets are useful for analyzing problems in a program running under the operating system. This program can be called any of the following:

- Installation-provided program
- An application program
- A non-authorized program
- A problem program
- A program in the private area

ABEND dumps are written for problems detected in two ways:

- **Software-detected problem**, such as:
  - A nonzero return code from a called module
  - A program check, abend code X'0Cx', that a recovery routine changes to another abend code

  - An erroneous control block queue
  - Not valid input to a system service
- **Hardware-detected problem**, which is a program check, abend code X'0Cx', that a recovery routine does not change to another abend code

Analysis Procedure

To analyze a SYSABEND or SYSUDUMP, take the following steps:

1. **Collect and analyze logrec error records.**

   Check all logrec error records related to the abended task. Determine if any records show an earlier system problem; if so, continue diagnosis with that problem. Because of recovery and percolation, a SYSABEND or SYSUDUMP dump can be the end result of an earlier system problem.
2. **Collect and analyze messages about the problem.** Use time stamps to select messages related to the problem:
   - The job log
   - The system log (SYSLOG) for the console with master authority

   Check the messages for earlier dumps written while the abended task was running. Determine if these earlier dumps indicate an earlier system problem; if so, continue diagnosis with that problem.
3. **Analyze the dump**, as described in the following steps.

   Note: After the problem and before the dump, recovery tried to reconstruct erroneous control block chains before ending the task. If the problem proves to be in a system component, a SYSABEND or SYSUDUMP dump cannot be used to isolate it because of the recovery actions; these dumps are useful only for problems in the private area.
4. **Obtain the abend code, reason code, job name, step name, and program status word (PSW) from the dump title** at the beginning of the dump.

   If the completion code is USER=dddd, an application program issued an ABEND macro to request the dump and to specify the completion code.

   If the completion code is SYSTEM=hhh, a system component ended the application program and a recovery routine in the program requested the dump. The application program probably caused the abend.

   Reference

   See *z/OS MVS System Codes* for an explanation of the abend code.
5. **Analyze the RTM2WA**, as follows:
   - In the TCB summary, find the task control block (TCB) for the failing task. This TCB has the abend code as its completion code in the CMP field. In the TCB summary, obtain the address of the recovery termination manager 2 (RTM2) work area (RTM2WA) for the TCB.
   - In the RTM2WA summary, obtain the registers at the time of the error and the name and address of the abending program.
   - If the RTM2WA summary does not give the abending program name and address, probably an SVC instruction abnormally ended.
   - If the RTM2WA summary gives a previous RTM2WA for recursion, the abend for this dump occurred while an ESTAE or other recovery routine was processing another, original abend. In recursive abends, more than one RTM2WA may be created. Use the previous RTM2WA to diagnose the original problem.

   References

- See *z/OS MVS Data Areas, Vol 4 (RD-SRRA)* for the RTM2WA and SDWA data areas.
- See *z/OS MVS Data Areas, Vol 5 (SSAG-XTLST)* for the TCB data area.

6. **Analyze the dump for the program name**. Obtain the program name from the RTM2WA summary. If the name field is zero, do the following:
   - Find the control blocks for the task being dumped.
   - The last request blocks are SVRBs. In the WLIC field in an SVRB, find the following SVC interruption codes:
     – X'33' for a SNAP SVC interruption
     – X'0C' for a SYNCH SVC interruption
   - The program request block (PRB) for the abending program immediately precedes these SVRBs.
   - When the dump contains more than one CDE, determine the first and last address for each CDE. The entry point address is the first address. Add the length to the entry point address to obtain the last address. Compare these addresses to the address in the right half of the PSW in the dump header; the PSW address falls between the first and last addresses of the correct CDE.

     Note that the leftmost digit in the PSW address denotes addressing mode and is not part of the address.
   - In that CDE, the NAME field gives the program name.
7. **Locate the failing program module** in the hexadecimal dump.
8. **Find the instruction that caused the abend.**

   The PSW in the dump header is from the time of the error. Obtain the address in the right half of the PSW. The leftmost digit denotes addressing mode and is not part of the address.

   For most problems, subtract the instruction length in the ILC field of the dump header from the PSW address to obtain the address of the failing instruction. Do not subtract the instruction length in the following cases; the failing instruction is at the PSW address.
   - Page translation exception.
   - Segment translation exception.
   - Vector operation interruption.
   - Other interruptions for which the processing of the instruction identified by the old PSW is nullified. See *z/Architecture Principles of Operation* for the interruption action.
   - If access registers were being used at the time of the error, so that the access list entry token (ALET) may be incorrect.

   Subtract the failing instruction address from the failing module address. Use this offset to find the matching instruction in the abending program’s assembler listing.
9. **For an abend from an SVC or system I/O routine, find the last program instruction.**

   If the abend occurred in a system component running on behalf of the dumped program, find the last instruction that ran in the program, as follows:
   - For an abend from an SVC routine, look in the last PRB in the control blocks for the task being dumped. The right half of the PSW in the RTPSW1 field contains the address of the instruction following the SVC instruction.

- For an abend from a system I/O routine, look in the save area trace. This trace gives the address of the I/O routine branched to. The return address in that save area is the last instruction that ran in the failing program.

10. **For an abend from an SVC or system I/O routine, determine the cause of the abend**, using the following:
    - For an abend from an SVC, look in the system trace table for SVC entries matching the SVRBs in the control blocks for the task being dumped.
    - For an abend from an I/O routine, look in the system trace table for I/O entries issued from addresses in the failing program. The addresses are in the PSW ADDRESS column.

    If SVC entries match the dumped blocks or the I/O entries were issued from the failing program, the system trace table was not overlaid between the problem and the dump.

    In this case, start with the most recent entries at the end of the trace. Back up to the last SVC entry with the TCB address of the abending task. Go toward the end of the trace, looking for indications of the problem. See Chapter 8, “System Trace” on page 8-1 for more information.

11. **For a program interrupt, determine the cause of the abend**, using the registers at the time of the error in the RTM2WA and in the SVRB following the PRB for the abending program.

    Also, look at the formatted save area trace for input to the failing module.

12. **For an abend in a cross memory environment**, do the following to analyze the dump.

    Many services are requested by use of the Program Call (PC) instruction, rather than by SVCs or SRBs. When an abend is issued by the PC routine, the OPSW field in the RB contains the instruction address of the PC routine that issued the abend. The SVRB contains the registers of the PC routine.

    Do the following to look for the registers and PSW at the time the PC instruction was issued:
    - For a stacking PC, find the registers in the linkage stack. Any entries on the linkage stack are before the RBs in the dump.
    - For a basic PC, find the registers in the PCLINK stack. Any entries on the PCLINK stack are after the RBs in the dump.

    For a stacking PC, find the linkage stack entry that corresponds to the RB/XSB for the program. The LSED field of the linkage stack entry and the XSBLSCP field in the corresponding XSB have the same value. From the linkage stack entry, obtain the registers and the PSW at the time the stacking PC was issued. The address in the PSW points to the instruction following the PC instruction in the abending program.

    For a basic PC, determine the caller from the PCLINK stack. To locate the PCLINK stack element (STKE):
    - The STKEs appear in the dump following all of the RBs. If the dump contains more than one STKE, the pointer to the STKE for the PC involved in the problem is in the XSBSTKE field of the XSB associated with the RB for the abending program.
    - The RBXSB field in the RB points to the XSB.
    - The XSBSEL field in the XSB points to the current STKE.

In the STKE, the STKERET field contains the return address of the caller of the PCLINK service.

Reference

See *z/OS MVS Data Areas, Vol 5 (SSAG-XTLST)* for the STKE and XSB data areas.

Chapter 6. SNAP Dumps

A SNAP dump is like getting a snapshot of yourself while hitting a baseball. You can go back later and look at what you did wrong so that you can improve.

Programming Interface information

This chapter (SNAP Dumps) contains programming interface information.

End of Programming Interface information

A SNAP dump shows virtual storage areas that a program, while running, requests the system to dump. A SNAP dump, therefore, is written while a program runs, rather than during abnormal end. The program can ask for a dump of as little as a one byte field to as much as all of the storage assigned to the current job step. The program can also ask for some system data in the dump.

A SNAP dump is especially useful when testing a program. A program can dump one or more fields repeatedly to let the programmer check intermediate steps in calculations. For example, if a program being developed produces incorrect results, requests for SNAP dumps can be added to the program to dump individual variables. The first time that incorrect storage is encountered should narrow down the section of code causing the error.

Major Topics

This chapter covers the following topics, which describe how to use SNAP dumps:
- “Obtaining SNAP Dumps”
- “Customizing SNAP Dump Contents” on page 6-4

Obtaining SNAP Dumps

Provide a data set to receive the dump, then arrange to print the dump. The SNAP or SNAPX macros in a program can place their dumps in the same or different data sets; the DCB parameter in each SNAP or SNAPX macro indicates the data set.

When setting up a dump data set, determine if the data set will contain privileged data. If so, protect it with passwords or other security measures to limit access to it.

Obtain a SNAP dump by taking the following steps:
1. Code a DD statement in the JCL for the job step that runs the problem program to be dumped with a ddname other than SYSUDUMP, SYSABEND, SYSMDUMP, or another restricted ddname. The statement can specify that the output of the SNAP dump should be written to one of the following:
   - Direct access.
   - Printer. Note that a printer is not recommended because the printer cannot be used for anything else while the job step is running, whether a dump is written or not.
   - SYSOUT. SNAP dumps usually use SYSOUT.
   - Tape.

Example: SYSOUT DD Statement for SNAP Dump

The following example places a SNAP dump in sysout output class A. In the example, output class A is a print class. When the system prints the output class, the system will print any dumps written to the class.

```
//SNAP1     DD  SYSOUT=A
```

Example: Tape DD Statement for SNAP Dump

The following example places a SNAP dump on a tape. In the example, TAPE is a group name established by the installation.

The system keeps the data set when the job step ends, whether normally or abnormally. In either case, SNAP dumps are taken throughout processing, regardless of the way the step ends.

```
//SNAP2     DD  DSN=DUMPDS,UNIT=TAPE,DISP=(,KEEP,KEEP)
```

Example: Direct Access DD Statement for SNAP Dump

The following example places a SNAP dump on direct access, for example, the 3350 direct access storage.

```
//SNAP3     DD  DSN=SNAPSHOT,UNIT=3350,DISP=(,KEEP,KEEP),
//              VOLUME=SER=12345,SPACE=(1680,(160,80))
```

The system writes the dump in a sequential data set using the basic sequential access method (BSAM). The dump data set can be on any device supported by BSAM.

2. In the problem program:
   a. Specify a data control block (DCB) for the data set to receive the dump. For a standard dump, which has 120 characters per line, the DCB must specify:
      BLKSIZE=882 or 1632
      DSORG=PS
      LRECL=125
      MACRF=(W)
      RECFM=VBA

      For a high-density dump, which has 204 characters per line and will be printed on an APA 3800 printer, the DCB must specify:
      BLKSIZE=1470 or 2724
      DSORG=PS
      LRECL=209
      MACRF=(W)
      RECFM=VBA
   b. Code an OPEN macro to open the DCB.

      Before you issue the SNAP or SNAPX macro, you must open the DCB that you designate on the DCB parameter, and ensure that the DCB is not closed until the macro returns control. To open the DCB, issue the DCB macro with the following parameters, and issue an OPEN macro for the data set:

```
DSORG=PS,RECFM=VBA,MACRF=(W),BLKSIZE=nnn,LRECL=xxx,
and DDNAME=any name but SYSABEND, SYSMDUMP or SYSUDUMP
```

If the system loader processes the program, the program must close the DCB after the last SNAP or SNAPX macro is issued.

c. Code a SNAP or SNAPX assembler macro to request the dump.

Example: Coding the SNAP Macro

In the following example, the SNAP macro requests a dump of a storage area, with the DCB address in register 3, a dump identifier of 245, the storage area's starting address in register 4, and the ending address in register 5:

```
     SNAP  DCB=(3),ID=245,STORAGE=((4),(5))
```

Repeat this macro in the program as many times as wanted, changing the dump identifier for a unique dump. The system writes all the dumps that specify the same DCB to the same data set.

Example: Two SNAP Dump Requests in a Program

The following example shows a problem program that requests two SNAP dumps. Both SNAP macros in the example specify the same data control block (DCB) to place both dumps in the same data set. Each dump has a different identifier: PIC3 for the first dump, PIC4 for the second. Both dumps show the same areas: the control blocks. Thus, the programmer can see these areas at two points in the program's processing.

```
SNAPDCB   DCB    BLKSIZE=882,DSORG=PS,LRECL=125,MACRF=(W),RECFM=VBA
           .
           .
           .
          OPEN   SNAPDCB,OUTPUT
          LA     3,SNAPDCB
          SNAP   DCB=(3),ID=PIC3,SDATA=CB
           .
           .
           .
          SNAP   DCB=(3),ID=PIC4,SDATA=CB
          CLOSE  SNAPDCB
```

d. Close the DCB with a CLOSE assembler macro.

References

- See *z/OS DFSMS Macro Instructions for Data Sets* for coding the DCB, OPEN, and CLOSE macros.

- See *z/OS MVS Programming: Assembler Services Reference ABE-HSP* and *z/OS MVS Programming: Assembler Services Reference IAR-XCT* for required parameters on the DCB macro and for coding the SNAP or SNAPX macro.

3. Print or view the data set. The output of the SNAP or SNAPX macro is a standard EBCDIC data set with ANSII characters in column one. This data set can be edited.

The dumps are formatted as they are created. Printing depends on the location of the dump when it is created:

Location — **Printing**

SYSOUT
The system prints the dump(s) when printing the output class.

On a tape or direct access data set
Print the dump(s) in a separate job or job step.

Printer
The system prints the dump(s) as they are created.

To view SNAP dumps at a terminal, browse the data set containing the dump.

Example: Printing a SNAP Dump

The following JCL prints a SNAP dump. Because the system formats the dump when creating it, the IEBPTPCH utility program can print the dump.

The dump is in the SNAPSHOT data set.

```
             .
             .
             .
//PRINT     EXEC  PGM=IEBPTPCH
//SYSPRINT  DD    SYSOUT=A
//SYSUT1    DD    DSN=SNAPSHOT,UNIT=3350,DISP=(OLD,DELETE),
//           VOLUME=SER=12345
//SYSUT2    DD    SYSOUT=A
//SYSIN     DD    *
   PRINT    TYPORG=PS
/*
```

Customizing SNAP Dump Contents

You can customize the contents of SNAP dumps in one of the following ways:
- Through installation exits.
- Through parameters on the SDUMP or SDUMPX macro.

Customizing through Installation Exits

An installation can customize the contents of SNAP dumps through the IEAVADFM or IEAVADUS installation exits. IEAVADFM is a list of installation routines to be run and IEAVADUS is one installation routine. The installation exit routine runs during control block formatting of a dump when the CB option is specified on the SNAP or SNAPX macro. The routine can format control blocks and send them to the data set for the dump.

Reference

See *z/OS MVS Installation Exits* for more information.

Customizing through the SNAP or SNAPX Macro

The parameters on the SNAP or SNAPX macro determine the dump contents. The macro can specify any or all of the areas listed in the following table.

Hiperspaces

Note that the parameters cannot request that a Hiperspace be included in the dump. To include Hiperspace data in a SNAP dump, read the data from the Hiperspace into address space storage that is being dumped.

Reference

See *z/OS MVS Programming: Extended Addressability Guide* for more information about manipulating data in Hiperspace storage.

Parameter	Dump Contents
ALLPA	All link pack areas, as follows: • Job pack area (JPA) • Link pack area (LPA) active for the task being dumped • Related supervisor call (SVC) modules
ALLVNUC	The entire virtual control program nucleus
CB	Control blocks for the task being dumped
DM	Data management control blocks for the task being dumped: • Data control block (DCB) • Data extent block (DEB) • Input/output block (IOB)
ERR	Recovery termination manager (RTM) control blocks for the task being dumped: • Extended error descriptor (EED) for RTM • Registers from the system diagnostic work area (SDWA) • RTM2 work area (RTM2WA) • Set task asynchronous exit (STAE) control block (SCB)
IO	Input/output supervisor (IOS) control blocks for the task being dumped: • Execute channel program debug area (EXCPD) • Unit control block (UCB)
JPA	Job pack area (JPA): module names and contents
LPA	Link pack area (LPA) active for the task being dumped: module names and contents
LSQA	Local system queue area (LSQA) allocated for the address space (that is, subpools 203 - 205, 213 - 215, 223 - 225, 229, 230, 233 - 235, 249, 253 - 255)
NUC	Read/write portion of the control program nucleus (that is, only non-page-protected areas of the DAT-on nucleus), including: • Communication vector table (CVT) • Local system queue area (LSQA) • Prefixed save area (PSA) • System queue area (SQA)
PCDATA	Program call information for the task
PSW	Program status word (PSW) when the dump is requested
Q	Global resource serialization control blocks for the task being dumped: • Global queue control blocks • Local queue control blocks

SNAP Dumps

Parameter	Dump Contents
REGS	Registers when the dump is requested: • Access registers • Floating-point registers • General registers • Vector registers, vector status register, and vector mask register for a task that uses the Vector Facility
SA or SAH	Save area linkage information, program call linkage information, and backward trace of save areas
SPLS	Storage allocated in user subpools 0 - 127, 129 -132, 244, 251, and 252 for the task being dumped
SQA	System queue area (SQA) allocated (that is, subpools 226, 239, 245, 247, 248)
SUBTASKS	Storage for the task being dumped and program data for all of its subtasks
SWA	Scheduler work area (SWA) (that is, subpools 236 and 237)
TRT	System trace and generalized trace facility (GTF) trace, as available
—	One or more data spaces identified on the SNAPX macro
—	One or more storage areas, identified by beginning and ending addresses on the SNAP or SNAPX macro
—	One or more subpools, identified by subpool number on the SNAP or SNAPX macro

Chapter 7. The Dump Grab Bag

Like that one drawer in your kitchen . . . full of all kinds of unrelated stuff that winds up being necessary when you least expect it.

A dump contains information about an error that can help you identify a problem type. Using IPCS, the information about the error can be formatted to provide a quick and effective method of retrieval.

The hints that follow apply to processing all kinds of dumps: SVC dumps, stand-alone dumps, and SYSMDUMP dumps.

Major Topics

This chapter covers the following topics:

- “Problem Data for Storage Overlays”
- “Problem Data from the Linkage Stack” on page 7-3
- “Problem Data for Modules” on page 7-4
- “Problem Data from Recovery Work Areas” on page 7-4
- “Problem Data for ACR” on page 7-5
- “Problem Data for Machine Checks” on page 7-6

Problem Data for Storage Overlays

When analyzing a dump you should always be aware of the possibility of a storage overlay. System problems in MVS are often caused by storage overlays that destroy data, control blocks, or executable code. The results of such an overlay vary. For example:

- The system detects an error and issues an abend code, yet the error can be isolated to an address space. Isolating the error is important in discovering whether the overlay is in global or local storage.
- Referencing the data or instructions can cause an immediate error such as a specification exception (abend X'0C4') or operation code exception (abend X'0C1').
- The bad data can be used to reference a second location, which then causes another error.

When you recognize that the contents of a storage location are not valid and subsequently recognize the bit pattern as a certain control block or piece of data, you generally can identify the erroneous process/component and start a detailed analysis.

Analyzing the Damaged Area

Once you determine that storage is bad or overlaid, try to identify the culprit. First, determine the extent of the bad data. Look for EBCDIC data or module addresses in storage to identify the owner. Any type of pattern in storage can indicate an error and identify the program that is using the damaged storage. Look at the data on both sides of the obviously bad areas. See if the length of the bad area is familiar; that is, can you relate the length to a known control block length, data size, MVC length? If so, check various offsets to determine their contents and, if you recognize some, try to determine the exact control block.

Example: Recognizing a Pattern

In the following output, storage from CSA shows a pattern of allocated blocks.

```
00CFD000   00000000   00000000   E5C7E3E3   080000F1   | ........VGTT...
00CFD010   00000020   00000000   0000E3D8   00000000   | ..........TQ...
00CFD020   00BE3D30   00000000   E5C7E3E3   080000F1   | ........VGTT...
00CFD030   00000020   00CFD008   0000E260   00000000   | ......}...S-...
00CFD040   00CA6A60   00000000   17000080   E5E2C90F   | ...-........VSI
00CFD050   00CFE018   00CFD0B8   C8E2D4D3   4BD4C3C4   | ..\...}.HSML.MC
00CFD060   E24BC4C1   E3C14040   40404040   40404040   | S.DATA
00CFD070 TO 00CFD07F (X'00000010' bytes)--All bytes contain X'40', C' '
00CFD080   40404040   000E0042   00000000   00000000   |     ..........
00CFD090   00000000   00000000   00000000   17CA0000   | ...............
00CFD0A0   1FFE0000   C2C3E3F3   D3C90001   00000000   | ....BCT3LI.....
00CFD0B0   00000000   40080000   E2E8E2F1   4BE4C3C1   | .... ...SYS1.UC
00CFD0C0   E34BC5D5   E5F2F600   00000168   00CFD0C8   | T.ENV26.......}
00CFD0D0   F1000000   00CFD230   00CFD22C   13C9C4C1   | 1.....K...K..ID
00CFD0E0   C8C5C240   00100150   00CFD244   00000160   | HEB ...&;.K....
```

Even if you do not recognize the pattern, take one more step. Can you determine the offset from some base that would have to be used in order to create the bit pattern? If so, the fact that there is a certain bit pattern at a certain offset can be helpful.

For example, a BALR register value (X'40D21C58') at an offset X'C' can indicate that a program is using this storage for a register save area (perhaps caused by a bad register 13). Another field in the same overlaid area might trigger recognition.

Repetition of a pattern can indicate a bad process. If you can recognize the bad data you might be able to relate that data to the component or module that is causing the error. This provides a starting point for further analysis.

Common Bad Addresses

The following are commonly known as bad addresses. If you recognize these in the code you are diagnosing, focus your problem source identification on these areas:

- X'000C0000', X'040C0000', or X'070C0000', and one of these addresses plus some offset. These are generally the result of some code using 0 as the base register for a control block and subsequently loading a pointer from 0 plus an offset, thereby picking up the first half of a PSW in the PSA.

  Look for storage overlays in code pointed to by an old PSW. These overlays result when 0 plus an offset cause the second half of a PSW to be used as a pointer.
- X'C00', X'D00', X'D20', X'D28', X'D40', and other pointers to fields in the normal FRR stack. Routines often lose the contents of a register during a SETFRR macro expansion and incorrectly use the address of the 24-byte work area returned from the expansion.
- Register save areas. Storage might be overlaid by code doing a store multiple (STM) instruction with a bad register save area address. In this case, the registers saved are often useful in determining the component or module at fault.

Problem Data from the Linkage Stack

The linkage stack can be used to identify a program that requested a system service, if the service was entered by a branch instruction.

Example: Viewing a Linkage Stack Entry

To see the linkage stack entry associated with address space identifier (ASID) X'1A', use the IPCS subcommand:

```
SUMMARY FORMAT ASID(X'1a')
```

The resulting dump for the linkage stack associated with the address space shows one entry, as follows:

```
LINKAGE STACK ENTRY  01  LSED: 7F7490B0
  LSE: 7F749010
     GENERAL PURPOSE REGISTER VALUES
     00-03.... 7FFEB410  04504DF4  04532000  04541FFF
     04-07.... 04504CE4  81150380  00000028  04504B50
     08-11.... 04503A75  04502A76  04501A77  04504630
     12-15.... 84500A78  00000000  80FD7618  8450FAF8
     ACCESS REGISTER VALUES
     00-03.... 00000000  00000000  00000000  00000000
     04-07.... 00000000  00000000  00000000  00000000
     08-11.... 00000000  00000000  00000000  00000000
     12-15.... 00000000  00000000  00000000  00000000
1S/A ON AQFT PER SYSTEM HUNG 22:30 08/30/88       24  09:42:48 10/14/88

PKM..8000 SASN..001A EAX..0000 PASN..001A PSW..070C0000  80FD7618
    TARG... 8450FB12  MSTA... 0451E300  00000000
    TYPE... 84
     BAKR STATE ENTRY
    RFS... 0F38      NES... 0000
```

BAKR STATE ENTRY
A Branch and Stack (BAKR) instruction caused this entry.

SASN..1A and PASN..1A
At the time of the BAKR, the program was not in cross memory mode. When the branching program is not in cross memory mode, secondary address space number (SASN) and primary address space (PASN) are identical. If the program had been in cross memory mode, SASN and PASN would not have been identical.

PSW..070C0000 80FD7618
The return address of the branch caused by the BAKR is FD7618. This address is in the right half of the program status word (PSW).

Many system services are called through branches. For branch entry services, use register 14 to identify the calling program. Look for the problem in the calling program.

See *z/OS MVS Programming: Extended Addressability Guide* for more information about the linkage stack.

Problem Data for Modules

For a module, the system saves and restores status from different locations, depending on the processing mode of the module when it lost control. Use the IPCS STATUS CPU subcommand to find out the mode of the module that had been currently running for each processor. Use the saved status as problem data for diagnosis.

Processing Modes

The processing modes follow. Code always runs in one or more of these modes. For example, code running in task or service request block (SRB) mode can also be either locally locked or physically disabled.

- **Task mode** is the most common processing mode. All programs given control by ATTACH, ATTACHX, LINK, LINKX, XCTL, and XCTLX macros run in task mode.
- **SRB mode** is code that runs from one of the service request block (SRB) queues.
- **Physically disabled mode** is reserved for high-priority system code that manipulates critical system queues and data areas. This mode is usually combined with supervisor state and key 0 in the PSW. The combination ensures that the routine can complete its function before losing control. The mode is restricted to just a few modules in the system, for example, interrupt handlers, the dispatcher, and programs holding a global spin lock.
- **Locked mode** is for code that runs in the system while holding a lock.
- **Cross memory mode**. Cross memory mode is defined by:
  - **Primary address space**: Address space identifier (ASID) in control register 3
  - **Secondary address space**: ASID in control register 4
  - **Home address space**: Address of the address space control block (ASCB) in the PSAAOLD field
  - **PSW S-bit** (bit 16 of the PSW): Indicator of current addressability:
    S-bit=0 - To the primary address space
    S-bit=1 - To the secondary address space

    When primary addressability and secondary addressability are to the home address space and the S-bit=0, the work is not in cross memory mode.
- **Access register (AR) mode**, where a program can use the full set of assembler instructions (except MVCP and MVCS) to manipulate data in another address space or in a data space. Unlike cross memory, access registers allow full access to data in many address spaces or data spaces.

Problem Data from Recovery Work Areas

You can use the recovery work area (RWA) to find the failing module. In most cases, you would use the TCB and RB structure to find the failing module instead of the RWA. Use the RWA in the following situations:

- When an SVC dump is requested in a SLIP trap. In this dump, the current status at the time of the problem is in the recovery save areas or in the SDUMP SQA 4K buffer. See “Reading the SDUMPX 4K SQA Buffer” on page 2-50 for more information.
- When the problem is in the recovery process itself.
- When a stand-alone dump is written because of a suspected loop.

The recovery work areas are:

- Logrec records
- Logrec buffer in the system: obtained by a VERBEXIT LOGDATA subcommand
- System diagnostic work area (SDWA), including the variable recording area (VRA): formatted in logrec records and in the logrec buffer.
- Functional recovery routine (FRR) stacks: described in the next topic.
- Recovery termination manager (RTM) data areas, including the RTM2 work area (RTM2WA): formatted by a SUMMARY FORMAT subcommand or obtained in a formatted ABEND or SNAP dump by the ERR option.

The RTM2WA and SDWA blocks contain registers, PSW, and other time of problem information. Use these blocks in diagnosis when they are associated with a task control block (TCB).

References

- See Chapter 14, “Recording Logrec Error Records” on page 14-1 for more information.
- See *z/OS MVS Data Areas, Vol 4 (RD-SRRA)* for the control blocks.
- See *z/OS MVS IPCS Commands* for the IPCS subcommands.

Problem Data for ACR

When alternate CPU recovery (ACR) is active at the time of the dump, the search argument in IPCS STATUS WORKSHEET output contains the symptom:

```
FLDS/CSDACR
```

Pre-Processing Phase Data

If ACR is active, problem data for the pre-processing phase are:

- The CSDCPUAL field of the common system data (CSD) indicates which processor failed and which is still running
- A system trace table entry with ACR in the IDENT column indicates that ACR was begun and identifies the failing processor
- Use the CSD online mask to determine which CPU’s LCCA to examine. Use the IPCS subcommand CBFORMAT to examine the failing CPU’s LCCA.
- The WSACACR in the CPU work save area vector table (WSAVTC) for both processors’ logical configuration communication areas (LCCA) points to a copy of the PSAs and FRR stacks for both processors.
- The LCCADCPU in both processors points to the LCCA of the failing processor and the LCCARCPU points to the LCCA of the running processor

Note that a dump shows the PSA of the failed processor when the running processor initiated ACR. The normal FRR stack, pointers to other FRR stacks, locks, PSA super bits, and other data reflect the processor at the time of the failure.

Post-Processing Phase Data

ACR issues message IEA858E when it completes and resets the CSDACR flag to X'00'.

Data Obtained by IPCS

Use the following IPCS subcommand to see all the LCCAs and the CSD:

```
STATUS CPU DATA WORKSHEET
```

References

- See *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)* for the control blocks.
- See *z/OS MVS IPCS Commands* for the IPCS subcommands.

Problem Data for Machine Checks

The hardware uses a machine check interruption to tell the control program that it has detected a hardware malfunction. Machine checks vary considerably in their impact on software processing:

- **Soft errors**: Some machine checks notify software that the processor detected and corrected a hardware problem that required no software recovery action.
- **Hard errors**: Other hardware problems detected by a processor require software-initiated action for damage repair. Hard errors also require software recovery to verify the integrity of the process that experienced the failure.

The machine check interrupt code (MCIC) in the PSA FLCMCIC field describes the error causing the interrupt. An MCIC can have more than one bit on to indicate more than one failing condition.

For a machine check, the system writes a logrec error record. The error record contains the MCIC, except when:

- The LRBMTCKS bit in field LRBMTERM of the logrec buffer (LRB) is ON to indicate that the machine check old PSW and the MCIC are both zero.
- The LRBMTINV bit in field LRBMTERM is ON to indicate that the machine check old PSW is nonzero but the MCIC is zero.

Hard errors cause FRR and ESTAE processing.

Reference

See *z/Architecture Principles of Operation* for a complete description of the MCIC.

Chapter 8. System Trace

A programmer's best friend - the system trace. Just like your dog, system trace is always there, running, and it doesn't need special care or feeding.

System trace provides an ongoing record of hardware and software events occurring during system initialization and operation. The system activates system tracing at initialization and the tracing runs continuously, unless your installation has changed the IBM-supplied system tracing. After system initialization, you can use the TRACE operator command on a console with master authority to customize system tracing.

Because system trace usually runs all the time, it is very useful for problem determination. While system trace and the general trace facility (GTF) lists many of the same system events, system trace also lists events occurring during system initialization, before GTF tracing can be started. System trace also traces branches and cross-memory instructions, which GTF cannot do.

Major Topics

The following topics explain system trace in detail:

- "Customizing System Tracing"
- "Receiving System Trace Data in a Dump" on page 8-2
- "Formatting System Trace Data in a Dump" on page 8-3
- "Reading System Trace Output" on page 8-3

Customizing System Tracing

The system starts system tracing during system initialization and the trace runs continually. There are, however, a few things you can do to alter system tracing:

- "Increasing the Size of the System Trace Table".
- "Tracing Branch Instructions" on page 8-2.

Increasing the Size of the System Trace Table

System trace tables reside in fixed storage on each processor. The default trace table size is 64 kilobytes per processor, but you can change it using the TRACE ST command. IBM does not recommend running with trace tables smaller than the default 64 kilobytes. You might, however, want to increase the size of the system trace table from the default 64 kilobytes when:

- You find that the system trace does not contain tracing from a long enough time period.
- You want to trace branch instructions (using the BR=ON option on the TRACE ST command when you start tracing).

Do the following to increase the size of the trace table:

- Enter the TRACE ST command to change the size of the system trace table. For example, to restart system tracing and increase the size of the trace table from the default 64 kilobytes per processor to 120 kilobytes per processor:

  ```
  TRACE ST,120K
  ```

Tracing Branch Instructions

System tracing allows you the option of tracing branch instructions, such as BALR, BASR, BASSM and BAKR, along with other system events. When the system is running in z/Architecture mode, the option of tracing branch instructions also includes mode tracing.

Attention: Branch tracing ON can affect your system performance and use very large amounts of storage. Do not use branch tracing as the default for system tracing on your system. You should only use it for short periods of time to solve a specific problem. The default system tracing does not include branch instructions.

When you want to trace branch instructions such as BALR, BASR, BASSM and BAKR, do the following:

- Restart system tracing with branch tracing using the TRACE command from a console with master authority:

  ```
  TRACE ST,BR=ON
  ```

  Because tracing branch instructions can significantly increase the number of trace entries being generated, you should increase the size of the trace tables from the default 64 kilobytes when you turn tracing on:

  ```
  TRACE ST,200K,BR=ON
  ```

Reference

- *Principles of Operation* describes the branch instruction trace entries and the mode trace entries that MVS combines with them (and are generated by the hardware). MVS enables or disables the production of these unformatted entries by manipulating control register bits by the instruction. The trace table entries that are not 'branch (or mode)' entries that are generated by MVS software through the TRACE or TRACG instructions. See the Tracing chapter for information.
- See the TTE Programming Interface Information chapter of the *z/OS MVS Data Areas, Vol 5 (SSAG-XTLST)* for a description of the TTE from mapping macro IHATTE.

Receiving System Trace Data in a Dump

System trace writes trace data in system trace tables in the trace address space. System trace maintains a trace table for each processor. Obtain the trace data in a dump that included option SDATA=TRT. The following table shows the dumps that have TRT in their default options and how to request trace data for dumps that do not include the data by default:

Dump	**How to Obtain Trace Data**
ABEND dump to SYSABEND	Default
ABEND dump to SYSMDUMP	Default
ABEND dump to SYSUDUMP	Default
SNAP dump	Request SDATA=TRT
Stand-alone dump	Default
SVC dump for SDUMP or SDUMPX macro	Default
SVC dump for DUMP operator command	Default

Dump	How to Obtain Trace Data
SVC dump for SLIP operator command with ACTION=SVCD, ACTION=STDUMP, ACTION=SYNCSVCD, or ACTION=TRDUMP	Default
Any dump customized to exclude trace data	Request SDATA=TRT

Formatting System Trace Data in a Dump

- For formatted dumps, system trace formats the system trace data and the system prints it directly.
- For unformatted dumps, use the IPCS SYSTRACE subcommand to format and print or view the trace data in the dump.

Reading System Trace Output

This topic describes system trace table entries (TTE) as they appear in a dump formatted with the IPCS SYSTRACE subcommand. The following topics appear:

- "Example of a System Trace in a Dump"
- "Summary of System Trace Entry Identifiers" on page 8-4 shows a table of the system trace identifiers for each system trace entry in a dump and shows where you can find the format of the entry in this section. If you are looking for a particular entry start with this table, because many of the entries are similar and so were grouped together.
- "ACR Trace Entries" on page 8-6 through "USRn Trace Entries" on page 8-29 shows the format for each type of trace entry. For the detailed format of TTEs, see *z/OS MVS Data Areas, Vol 5 (SSAG-XTLST)*.

Example of a System Trace in a Dump

The following example shows system trace entries. IPCS formatted the entries from an example SVC dump. Note that system trace data in an ABEND dump has the same format. The subcommand issued from the IPCS Subcommand Entry panel was:

```
SYSTRACE
```

The oldest trace entries appear first in the trace; the newest entries are at the end. An asterisk (*) before an identifier indicates an unusual condition; see the format of the entry for an explanation.

System Trace

```
-------------------------------------------------- SYSTEM TRACE TABLE ----------------------------------------------
--                                                 ------------------                                              --
--                                                 ------------------                                              --
PR ASID  WU-ADDR  IDENT CD/D PSW----- ADDRESS-  UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6

00 0034 009FF0B0   DSP       070C0000 815FCB40  00800000 00000000 021957E0  00000001 00000000 0034 0034 B5ACFF8810B4CE88
00 0034 009FF0B0   PC    ...  0       01259B96           0030A
00 0034 009FF0B0   PR    ...  0       01259B96 012221E4                                       0034
00 0034 009FF0B0   PC    ...  0       01259B96           0030A
00 0034 009FF0B0   PR    ...  0       01259B96 012221E4                                       0034
01 000A 009F76C0   SVC      1 07041000 0001DC4A  00000000 00000001 FF03E7B8                                B5ACFF8810B59C84
01 000A 009F76C0   SVCR     1 070C1000 0001DC4A  809F5BD8 00000001 FF03E7B8                                B5ACFF8810B5CD24
01 000A 009FF458   DSP       070C0000 90824A0A  00000000 108240C6 1082C1F8  00000001 00000000 000A 000A B5ACFF8810B5ED64
01 000A 009FF458   SVC      2 07041000 90823946  00000001 00000000 00FC16D8                                B5ACFF8810B6BA44
01 000A 009FF458   SVCR     2 070C1000 90823946  00FF00FF 00000002 00FF00FF                                B5ACFF8810B6E704
01 000A 009FF458   PC    ...  0       108224A2           00408
01 000A 009FF458   PC    ...  0       1080031A           0030B
00 0034 009FF0B0   CALL      07040000 815FCB40  00011202 00000000           00000001 00000000 0034 0034 B5ACFF8810B6F288
                                                                            00000000
00 000A 009F76C0   DSP       070C1000 0001DC4A  00000000 00000001 FF03E7B8  00000000 00000000 000A 000A B5ACFF8810B72068
00 000A 009F76C0   SUSP               0001E902  009F5BD8 LOCL     00000000  00000000 00000000           B5ACFF8810B73448
                                                00000000                    00000000
00 0034 009FF0B0   DSP       070C0000 815FCB40  00800000 00000000 021957E0  00000001 00000000 0034 0034 B5ACFF8810B74B48
00 0034 009FF0B0   PC    ...  0       01259B96           0030A
00 0034 009FF0B0   PR    ...  0       01259B96 012221E4                                       0034
01 000A 009FF458   SSRV  132          00000000  0000E632 00001000 7FFA7000                                B5ACFF8810B8CA44
                                                000A0000
01 000A 009FF458   PR    ...  0       1080031A 012A3A36                                       000A
01 000A 009FF458   PC    ...  0       10800388           00311
01 000A 009FF458   CALL      07041000 812A3310  00001202 00000000           00000001 00000000 000A 000A B5ACFF8810B96464
                                                                            00000000
01 000A 009FF458   DSP       070C1000 812A3310  00800000 80000000 10800388  00000001 00000000 000A 000A B5ACFF8810B9A004
01 000A 009FF458   SSRV  133          00000000  0000E603 00001000 7FFA7000                                B5ACFF8810BAA784
                                                000A0000
01 000A 009FF458   PR    ...  0       10800388 012A3A36                                       000A
01 000A 009FF458   PR    ...  0       108224A2 108001D0                                       000A
01 000A 009FF458   SVC      1 07041000 9082217A  809F5108 00000001 EF7D36D0                                B5ACFF8810BADC84
01 000A 009FF458   SVCR     1 070C1000 9082217A  809F5108 00000001 EF7D36D0                                B5ACFF8810BAEA04
01 000A 009F76C0   DSP       070C0000 0001E902  00000000 00000001 FF03E7B8  00000001 00000000 000A 000A B5ACFF8810BB0284
01 000A 009F76C0   SVC      1 07041000 0001DC4A  0001B4B0 00000001 FF03E7B8                                B5ACFF8810BBAC64
01 000A 009F76C0   SVCR     1 070C1000 0001DC4A  809F5BD8 00000001 FF03E7B8                                B5ACFF8810BBD944
01 0001 00000000   WAIT                                                                                   B5ACFF8810BBF384
00 0034 009FF0B0   EXT   1005 07040000 8620CC68  00001005                    00000001 00000000 0034 0034 B5ACFF8810BF7A08
                                                                            00000000
00 0034 009FF0B0   SSRV  112          81257294  01FC72D8 00F42800 87D1B410                                B5ACFF8810C39108
                                                00000000
00 0034 009FF0B0   SVCR    33 071C2000 8015CE86  00000000 00000001 8015CDA0                                B5ACFF8810C44048
00 0005 061AB238   SRB       070C0000 87D1B410  00000005 01FC72D8 00000000        00          0005 0005 B5ACFF8810C47108
                                                009FF968 00
```

Summary of System Trace Entry Identifiers

This topic summarizes all the system trace entries by identifier. Because many trace entries are similar, they are described together. Use the table below to locate the format for a particular entry.

Example: Finding the format for an SVC entry

In the following trace entry, the system trace identifier is SVC:

```
01 000C 00AFF090   SVC    1 070C2000 00EB19CC  00000000 00000001 00C13340
```

Look up SVC in Table 8-1 to find the page where the SVC trace entry format is described. In this case, the SVC trace entry is described in “SVC, SVCE, and SVCR Trace Entries” on page 8-24.

Table 8-1. References For System Trace Entry Format Description

Identifier (IDENT)	Description	For format, see:
ACR	Alternate CPU recovery	"ACR Trace Entries" on page 8-6
ALTR	Alteration of trace option	"ALTR Trace Entries" on page 8-7
BR	Branch through a BAKR, BALR, BASR, or BASSM instruction	"BR Trace Entries" on page 8-8
BSG	Branch on subspace group	"BSG, PC, PR, PT, and SSAR Trace Entries" on page 8-9
CALL	External call external interruption	"CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries" on page 8-15
CLKC	Clock comparator external interruption	"CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries" on page 8-15
CSCH	Clear subchannel operation	"CSCH, HSCH, MSCH, RSCH, SSCH and SIGA Trace Entries" on page 8-13
DSP	Task dispatch	"DSP, SRB, SSRB, and WAIT Trace Entries" on page 8-12
EMS	Emergency signal external interruption	"CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries" on page 8-15
EXT	General external interruption	"CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries" on page 8-15
HSCH	Halt subchannel operation	"CSCH, HSCH, MSCH, RSCH, SSCH and SIGA Trace Entries" on page 8-13
I/O	Input/output interruption	"CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries" on page 8-15
MCH	Machine check interruption	"CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries" on page 8-15
MOBR	Change of addressing mode along with a change of instruction address	"MODE and MOBR Trace Entries" on page 8-11
MODE	Change of addressing mode	"MODE and MOBR Trace Entries" on page 8-11
MSCH	Modify subchannel operation	"CSCH, HSCH, MSCH, RSCH, SSCH and SIGA Trace Entries" on page 8-13
PC	Program Call control instruction	"BSG, PC, PR, PT, and SSAR Trace Entries" on page 8-9
PGM	Program interruption	"PGM and SPER Trace Entries" on page 8-19
PR	Program Return control instruction	"BSG, PC, PR, PT, and SSAR Trace Entries" on page 8-9
PT	Program Transfer control instruction	"BSG, PC, PR, PT, and SSAR Trace Entries" on page 8-9
RCVY	Recovery event	"RCVY Trace Entries" on page 8-20
RSCH	Resume subchannel operation	"CSCH, HSCH, MSCH, RSCH, SSCH and SIGA Trace Entries" on page 8-13
RST	Restart interruption	"CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries" on page 8-15
SIGA	Signal adapter operation	"CSCH, HSCH, MSCH, RSCH, SSCH and SIGA Trace Entries" on page 8-13
SPER	SLIP program event recording	"PGM and SPER Trace Entries" on page 8-19

Table 8-1. References For System Trace Entry Format Description (continued)

Identifier (IDENT)	Description	For format, see:
SRB	Initial service request block dispatch	"DSP, SRB, SSRB, and WAIT Trace Entries" on page 8-12
SS	Service signal external interruption	"CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries" on page 8-15
SSAR	Set Secondary Address Space Number control instruction	"BSG, PC, PR, PT, and SSAR Trace Entries" on page 8-9
SSCH	Start subchannel operation	"CSCH, HSCH, MSCH, RSCH, SSCH and SIGA Trace Entries" on page 8-13
SSRB	Suspended service request block dispatch	"DSP, SRB, SSRB, and WAIT Trace Entries" on page 8-12
SSRV	System service entered by a Program Call (PC) instruction or a branch	"SSRV Trace Entries" on page 8-25
SUSP	Lock suspension	"SUSP Trace Entries" on page 8-17
SVC	Supervisor call interruption	"SVC, SVCE, and SVCR Trace Entries" on page 8-24
SVCE	SVC error	"SVC, SVCE, and SVCR Trace Entries" on page 8-24
SVCR	SVC return	"SVC, SVCE, and SVCR Trace Entries" on page 8-24
TIME	Timer services	"TIME Trace Entries" on page 8-28
USRn	User event	"USRn Trace Entries" on page 8-29
WAIT	Wait task dispatch	"DSP, SRB, SSRB, and WAIT Trace Entries" on page 8-12
?EXPL	The SYSTRACE subcommand cannot identify the system trace entry	N/A

ACR Trace Entries

Purpose

An ACR trace entry represents failure of a processor and subsequent entry into the alternate CPU recovery component.

Entry Format

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS-  PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE-

pr fail tcb-addr *ACR   cpu                    psaeepsw flg-crex psacstk-  psaclhs-  psalocal           timestamp-------
                                               psasuper psamodew           psaclhse-
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`fail`: Home address space identifier (ASID) of the failing processor

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task for which the TTE was produced.

IDENT
The TTE identifier, as follows:

ACR Alternate CPU recovery

An asterisk (*) always appears before ACR to indicate an unusual condition.

CD/D
`cpu`: The failing processor address from the PSACPUPA field of the PSA

PSW----- ADDRESS-
Blank

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
`flg-crex`: LCCACREX field of the logical configuration communication area (LCCA) for the failing processor.
`psacstk-`: PSACSTK field from the prefix save area (PSA) from the failing processor.
`psaeepsw`: PSAEEPSW field in the PSA. Bytes 1 and 2 contain the failing processor's address. Bytes 3 and 4 contain the external interruption code.
`psamodew`: PSAMODEW field in the PSA
`psasuper`: PSASUPER field in the PSA

PSACLHS-
`psaclhs-`: String for the current lock held, from the PSACLHS field of the PSA from the failing processor.

PSACLHSE-
`psaclhse-`: Extended string for the current lock held, from the PSACLHSE field of the PSA from the failing processor.

PSALOCAL
`psalocal`: Locally locked address space indicator, from the PSALOCAL field of the PSA from the failing processor.

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on logrec data set records.

ALTR Trace Entries

Purpose

An ALTR trace entry represents alteration of the system trace options. The options are altered by a TRACE ST operator command.

Entry Format

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

pr home tcb-addr *ALTR                         tobtropt gpr0---- gpr1----                   pasd sasd timestamp-------
                                               pol-buf-
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`home`: Home address space identifier (ASID) associated with the TTE.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task or the work element block WEB).

IDENT
The TTE identifier, as follows:

ALTR Alteration of the trace option

An asterisk (*) always appears before ALTR to indicate an unusual condition.

CD/D
Blank

PSW----- ADDRESS-
Blank

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
`tobtropt`: Trace options in control register 12 format, from the TOBTROPT field of the system trace option block (TOB)
`gpr0----`: General register 0
`gpr1----`: General register 1
`pol-`: The number of processor with tracing active or suspended, from the TOBTRPOL field of the TOB
`buf-`: The number of trace buffers per processor, from the TOBTRBUF field of the TOB

PSACLHS-
Blank

PSALOCAL
Blank

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on logrec data set records.

BR Trace Entries

Purpose

A BR trace entry represents processing of a Branch and Link (BALR), Branch and Save (BASR), Branch and Save and Set Mode (BASSM), or Branch and Stack (BAKR) instruction, when the R_2 field in the instruction is not zero. These branches are traced only when a TRACE operator command requests branch tracing by BR=ON.

Entry Format

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

pr last tcb-addr  BR       address- address-  address- address- address- address- etc.
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`last`: Last home address space identifier (ASID) in the trace buffer.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task for which the TTE was produced.

IDENT
The TTE identifier, as follows:

BR Branch instruction

CD/D
Blank

PSW----- ADDRESS-
UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
PSACLHS-
PSALOCAL
PASD
SASD
TIMESTAMP-RECORD
`address-`: Successful branch address, repeated for consecutive branches on the BR entry. Addresses appear in the following formats:

Addressing mode and location	Appearance
24-bit address	xxxxxx
31-bit address	xxxxxxxx
64-bit address with zeros in high order bits	00_xxxxxxxx
64-bit address with non-zero high order bits	xxxxxxxx_xxxxxxxx

BSG, PC, PR, PT, and SSAR Trace Entries

Purpose

These trace entries represent processing of a cross memory instruction:
- A BSG trace entry represents a Branch on Subspace group (BSG) control instruction
- A PC trace entry represents a Program Call (PC) control instruction
- A PR trace entry represents a Program Return (PR) control instruction
- A PT trace entry represents a Program Transfer (PT) control instruction
- An SSAR trace entry represents a Set Second Address Space Number (SSAR) control instruction

Entry Formats

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

pr last tcb-addr  PC         psw-key- pc-addr-          pc#-----                            sasd

pr last tcb-addr  PR         psw-key- pr-addr-          pr-faddr                       pasd

pr last tcb-addr  PT         psw-key- pt-addr-          pt-asid-                       pasd sasd

pr last tcb-addr  SSAR       newsasid                                                       sasd

pr last tcb-addr  BSG        alet     bsg-addr
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`last`: Last home address space identifier (ASID) associated with the TTE.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task for which the TTE was produced.

IDENT
The TTE identifier, as follows:

PC	Program Call control instruction
PR	Program Return control instruction
PT	Program Transfer control instruction
SSAR	Set Secondary Address Space Number control instruction
BSG	Branch on Subspace Group control instruction

CD/D
Blank

PSW----- ADDRESS-
`alet`: ALET word during BSG execution
`newsasid`: New SASID from the SSAR instruction
`return--`: Caller's return address
`psw-key-`: Program status word (PSW) key
`pc-addr-`: Return address from the PC instruction
`pr-addr-`: New instruction address as updated by the PR instruction
`pt-addr-`: New instruction address as updated by the PT instruction
`bsg-addr`: New instruction address as updated by the BSG instruction

Addresses appear in the following formats:

Addressing mode and location	Appearance
24-bit address	xxxxxx
31-bit address	xxxxxxxx
64-bit address with zeros in high order bits	00_xxxxxxxx
64-bit address with non-zero high order bits	xxxxxxxx_xxxxxxxx

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
`pc#-----`: PC number from the PC instruction
`pr-faddr`: Address of the location following the PR instruction
`pt-asid-`: New ASID specified on the PT instruction

PSACLHS-
Blank

PSALOCAL
Blank

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
Blank

MODE and MOBR Trace Entries

Purpose

These trace entries represent a change of addressing mode:
- A MODE trace entry represents a change into or out of 64-bit addressing mode
- A MOBR trace entry represents a change into or out of 64-bit addressing mode along with a change of instruction address

Entry Format

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

pr last tcb-addr  MODE      target   address-  address- address- address- address- etc.

pr last tcb-addr  MOBR      target   address-  address- address- address- address- etc.
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`last`: Last home address space identifier (ASID) in the trace buffer.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task for which the TTE was produced.

IDENT
The TTE identifier, as follows:

MODE Addressing mode change instruction

MOBR Addressing mode change combined with a branch instruction

CD/D
Blank

PSW-----
`target`: Target addressing mode.

24 OR 31
Target addressing mode is either 24-bit or 31-bit.

64 Target addressing mode is either 64-bit.

ADDRESS-
UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
PSACLHS-
PSALOCAL
PASD

SASD
TIMESTAMP-RECORD
`address-`: Target address. Addresses appear in the following formats:

Addressing mode and location	Appearance
24-bit address	xxxxxx
31-bit address	xxxxxxxx
64-bit address with zeros in high order bits	00_xxxxxxxx
64-bit address with non-zero high order bits	xxxxxxxx_xxxxxxxx

DSP, SRB, SSRB, and WAIT Trace Entries

Purpose

These trace entries represent the dispatch of a unit of work:
- A DSP trace entry represents dispatch of a task
- An SRB trace entry represents the initial dispatch of a service request
- An SSRB trace entry represents dispatch of a suspended service request
- A WAIT trace entry represents dispatch of the wait task

Entry Formats

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

pr home  wu-addr  DSP       dsp-new- psw-----  psamodew gpr0---- gpr1----  psaclhs- psalocal pasd sasd timestamp-------

pr home  wu-addr  SRB       srb-new- psw-----  safnasid gpr0---- gpr1----  srbhlhi-          pasd sasd timestamp-------
                                               purgetcb flg-srb-

pr home  wu-addr  SSRB      ssrb-new psw-----  safnasid          gpr1----  psaclhs4 psalocal pasd sasd timestamp-------

pr home  wu-addr  WAIT                                                                                timestamp-------
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`home`: Home address space identifier (ASID) associated with the TTE.

WU-ADDR
`wu-addr`: Address of the task control block (TCB) for the current task or the work element block (WEB).

IDENT
The TTE identifier, as follows:
DSP Task dispatch
SRB Initial service request dispatch
SSRB Suspended service request dispatch
WAIT Wait task dispatch

CD/D
Blank

PSW----- ADDRESS-
`dsp-new- psw`: Program status word (PSW) to be dispatched
`srb-new- psw`: PSW to receive control on the SRB dispatch
`ssrb-new psw`: PSW to receive control on the SSRB redispatch

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6

`gpr0----`: General register 0
`gpr1----`: General register 1
`psamodew`: PSAMODEW field in the PSA
`safnasid`: LCCASAFN field in the logical configuration communication area (LCCA) and the related ASID
`flg-srb`: SRBFLGS field from the SRB
`purgetcb`: TCB (located in address space of the scheduler of the SRB) that gets control if the SRB abends and percolates

PSACLHS-
`psaclhs-`: String for the current lock held, from the PSACLHS field of the PSA.
`psaclhs4`: PSACLHS4 field of the PSA
`srbhlhi-`: SRBHLHI field in the SRB

PSALOCAL
`psalocal`: Locally locked address space indicator, from the PSALOCAL field of the PSA.

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on logrec data set records.

CSCH, HSCH, MSCH, RSCH, SSCH and SIGA Trace Entries

Purpose

These trace entries represent an input/output operation:
- A CSCH trace entry represents a clear subchannel operation
- An HSCH trace entry represents a halt subchannel operation
- An MSCH trace entry represents a modify subchannel operation
- An RSCH trace entry represents a resume subchannel operation
- An SSCH trace entry represents a start subchannel operation
- An SIGA trace entry represents a signal adapter operation

Entry Formats

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

pr asid tcb-addr  CSCH  dev cc  di   iosbaddr  ucb-addr ioq-addr asc-iosb                              timestamp-------

pr asid tcb-addr  HSCH  dev cc  di   iosbaddr  ucb-addr ioq-addr asc-iosb                              timestamp-------

pr asid tcb-addr  MSCH  dev cc       iosbaddr  ucb-addr f1f2pmom mbi-t2lb                              timestamp-------

pr asid tcb-addr  RSCH  dev cc  di   iosbaddr  ucb-addr                                                timestamp-------

pr asid tcb-addr  SSCH  dev cc  di   iosbaddr  ucb-addr orb-wrd2 orb-wrd3                              timestamp-------
                                               orb-wrd4 cap-addr bdev
 pr asid tcb-addr  SIGA  dev cc  fc   qib-addr  subsysid q-mask-1 q-mask-2                              timestamp-------
                                                         ucb-addr
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`asid`: Address space identifier (ASID) related to the I/O.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task or the work element block WEB).

IDENT
The TTE identifier, as follows:

CSCH	Clear subchannel operation
HSCH	Halt subchannel operation
MSCH	Modify subchannel operation
RSCH	Resume subchannel operation
SSCH	Start subchannel operation
SIGA	Signal adapter operation

An asterisk before RSCH, SSCH, or SIGA indicates that the condition code associated with the I/O was not 0.

CD/D
`dev`: One of the following:
- The device number associated with the I/O
- ADMF, if the IOSADMF macro was transferring data

PSW----- ADDRESS-
`cc`: Condition code in bits 2 and 3 associated with the I/O
`di`: Driver identifier associated with the I/O
`fc`: Function code associated with the I/O
`iosbaddr`: I/O supervisor block (IOSB) address associated with the I/O
`qib-addr`: Queue identification block (QIB) address associated with the I/O

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
`asc-iosb`: IOSB address for the associated SSCH request for the I/O
`bdev`: The base device number if the I/O is associated with an alias device.
`cap-addr`: Captured unit control block (UCB) address associated with the SSCH I/O. This field is blank if a below 16 megabyte UCB or actual above 16 megabyte UCB address was used for the start subchannel (SSCH) operation. The address of the actual above 16 megabyte UCB is in the `ucb-addr` field.
`f1f2pmom`: From the subchannel information block (SCHIB) associated with the I/O, as follows:

f1 SCHFLG1 flag field
f2 SCHFLG2 flag field
pm SCHLPM field
om SCHPOM field
`ioq-addr`: I/O queue (IOQ) address associated with the I/O
`mbi-t2lb`:
mbi- SCHMBI field from the SCHIB
t2 IOSOPT2 field from the IOSB
lb IOSFLB field from the IOSB
`orb-wrd2`: Word 2 of the operation request block (ORB) associated with the I/O
`orb-wrd3`: Word 3 of the operation request block (ORB) associated with the I/O
`orb-wrd4`: Word 4 of the operation request block (ORB) associated with the I/O
`q-mask-1`: Read or write queue mask associated with the I/O
`q-mask-2`: Read queue mask associated with the I/O
`subsysid`: Subsystem ID associated with the I/O
`ucb-addr`: Unit control block (UCB) address associated with the I/O

PSACLHS-
Blank

PSALOCAL
Blank

PASD
Blank

SASD
Blank

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on the logrec data set records.

CALL, CLKC, EMS, EXT, I/O, MCH, RST, and SS Trace Entries

Purpose

These trace entries represent an interruption:

- Five of the entries represent external interruptions:
  - A CALL trace entry is for an external call
  - A CLKC trace entry is for a clock comparator
  - An EMS trace entry is for an emergency signal
  - An EXT trace entry is for a general external interruption
  - An SS trace entry is for a service signal
- An I/O trace entry is for an I/O interruption
- An MCH trace entry is for a machine check
- An RST trace entry is for a restart

Entry Formats

```
PR ASID   WU-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS-  PSALOCAL PASD SASD TIMESTAMP-RECORD
                                                UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE-

pr home tcb-addr  CALL      ext-old- psw-----  psaeepsw tge-tcb- tge-asid  psaclhs-  psalocal pasd sasd timestamp-------
                                                                           psaclhse-

pr home tcb-addr  CLKC      ext-old- psw-----  psaeepsw tge-tcb- tge-asid  psaclhs-  psalocal pasd sasd timestamp-------
                                                                           psaclhse-
                                                 pccaemse

pr home tcb-addr  EMS       ext-old- psw-----  psaeepsw pccaemsi pccaemsp  psaclhs- psalocal pasd sasd timestamp-------
                                                 pccaemse

pr home tcb-addr  EXT  code ext-old- psw-----  psaeepsw                    psaclhs-  psalocal pasd sasd timestamp-------
                                                                           psaclhse-

pr home tcb-addr  I/O   dev i/o-old- psw-----  flg-ctl- ccw-addr dvch-cnt  psaclhs-  psalocal pasd sasd timestamp-------
                                                        ucb-addr ext-stat  psaclhse-
pr home tcb-addr *MCH       mch-old- psw-----  machine- chk-code psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                                                           psaclhse-
pr home tcb-addr *RST       rst-old- psw-----  gpr15--- gpr0---- gpr1----  psaclhs- psalocal pasd sasd timestamp-------
                                                 psasuper psamodew

pr home tcb-addr  SS        ext-old- psw-----  psaeepsw psaeparm msf-bcmd  psaclhs-  psalocal pasd sasd timestamp-------
                                                flg-brsp mssfasid mssfatcb psaclhse-
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`home`: Home address space identifier (ASID) associated with the TTE.

WU-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task or the work element block WEB).

IDENT
The TTE identifier, as follows:

CALL	External call external interruption
CLCK	Clock comparator external interruption
EMS	Emergency signal external interruption
EXT	General external interruption
I/O	I/O interruption
MCH	Machine check interruption
RST	Restart interruption
SS	Service signal external interruption

An asterisk (*) always appears before MCH and RST to indicate an unusual condition.

An asterisk before EXT indicates that the interrupt is a malfunction alert (MFA) or is the result of pressing the External Interrupt key.

An asterisk before I/O indicates that one of the following bits in IRBFLAGS field of the interrupt request block (IRB) is ON. The IRBFLAGS field is in the UNIQUE-1 column of the I/O entry.
- IRBN for path not operational
- IRBSALRT for alert status

CD/D
`code`: External interruption code
`dev`: Device number associated with the I/O or, for a co-processor device, the I/O co-processor identifier, for example, ADM

PSW----- ADDRESS-
`ext-old- psw`: External old program status word (PSW)

i/o-old- psw: I/O old PSW
mch-old- psw: Machine check old PSW
rst-old- psw: Restart old PSW

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6

ccw-addr: Address of the channel command word (CCW) for the I/O
-cnt: Residual count
dvch: Device status and subchannel status
ext-stat: Extended status word
flg-brsp: Maintenance and service support facility (MSSF) hardware flags and MSSF response code
flg-ctl-: IRBFLAGS field in the IRB and the subchannel control bytes
gpr15--- gpr0---- gpr1----: General registers 15, 0, and 1
machine- chk-code: Machine check interruption code from the FLCMCIC filed in the prefix save area (PSA)
msf-bcmd: Service processor command word
mssfasid: Service processor address space ID
mssfatch: Service processor TCB address
pccaemse: PCCAEMSE field from the physical configuration communication area (PCCA)
pccaemsi: PCCAEMSI field from the PCCA
pccaemsp: PCCAEMSP field from the PCCA
pccarph-: PCCARPB field from the PCCA
psaeepsw: PSAEEPSW field in the PSA. For CALL and EMS, bytes 1 and 2 contain the issuing processor's address. For all entries, bytes 3 and 4 contain the external interruption code.
psaeparm: PSAEPARM field in the PSA, containing the MSSF buffer address
psamodew: PSAMODEW field in the PSA
psasuper: PSASUPER field in the PSA
tge-asid: ASID of the associated timer queue element (TQE)
tge-tcb-: Address in the TCB for the associated TQE
ucb-addr: Unit control block (UCB) address

PSACLHS-
psaclhs-: String for the current lock held, from the PSACLHS field of the PSA.

PSACLHSE-
psaclhse-: Extended string for the current lock held, from the PSACLHSE field of the PSA.

PSALOCAL
psalocal: Locally locked address space indicator, from the PSALOCAL field of the PSA.

PASD
cpsd: Primary ASID (PASID) at trace entry.

SASD
sasd: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
timestamp-------: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on the logrec data set records.

SUSP Trace Entries

Purpose

An SUSP trace entry represents a request for a suspend type lock when the requestor had to be suspended because the lock was not available.

Entry Format

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS-  PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE-

 pr home tcb-addr  SUSP               return--  rb-addr- suspndid rel-addr  psaclhs-  psalocal          timestamp-------
                                                ssrbaddr                    psaclhse-
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`home`: Home address space identifier (ASID) associated with the TTE.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task

IDENT
The TTE identifier, as follows:

SUSP   Lock suspension

CD/D
Blank

PSW----- ADDRESS-
`return--`: Caller’s return address

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
`rb-addr-`: Address of the suspended request block (RB)
`rel-addr`: Address associated with the type of lock suspension:
- 0: for LOCL lock
- ASCB address: for CML lock
- Lockword address: for CEDQ, CMS, and CSMF locks

`ssrbaddr`: Address of the suspended service request block (SSRB)
`suspndid`: Identifier of the lock suspension type: CEDQ, CML, CMS, CSMF, or LOCL

PSACLHS-
`psaclhs-`: String for the current lock held, from the PSACLHS field of the PSA.

PSACLHSE-
`psaclhse-`: Extended string for the current lock held, from the PSACLHSE field of the PSA.

PSALOCAL
`psalocal`: Locally locked address space indicator, from the PSALOCAL field of the PSA.

PASD
Blank

SASD
Blank

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on the logrec data set records.

PGM and SPER Trace Entries

Purpose

These trace entries represent a program event:
- A PGM trace entry is for a program interrupt
- An SPER trace entry is for a PER event requested in a SLIP trap

Entry Formats

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS-  PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE-

pr home tcb-addr  PGM  code pgm-old- psw-----  ilc-code tea-----           psaclhs-  psalocal pasd sasd timestamp-------
                                                        tea-----           psaclhse-
pr home tcb-addr  SPER code pgm-old- psw-----  ilc-code          trap----  psaclhs-  psalocal pasd sasd timestamp-------
                                                per-addH per-addL           psaclhse-
```

PR
: `pr`: Identifier of the processor that produced the TTE.

ASID
: `home`: Home address space identifier (ASID) associated with the TTE.

TCB-ADDR
: `tcb-addr`: Address of the task control block (TCB) for the current task or the work element block (WEB).

IDENT
: The TTE identifier, as follows:
PGM Program interruption
SPER SLIP program event recording

An asterisk (*) before PGM indicates an unusual condition. PGM trace entries for program interrupts that may be resolved are not flagged. If the program interrupt is not resolved, then a subsequent RCVY trace entry is created and is flagged with an asterisk.

CD/D
: `code` for PGM entry: Program interruption code
`code` for SPER entry: PER number

PSW----- ADDRESS-
: `pgm-old- psw`: Program old program status word (PSW)

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
: `ilc-code`: Instruction length code and interruption code
`per-addH`: high order bits of the SLIP/PER status address
`per-addL`: low order bits of the SLIP/PER status address
`tea-----`: Translation exception address. In the high-order bit, 0 indicates primary and 1 indicates secondary.
`trap----`: SLIP/PER trap identifier in the form ID=xxxx

PSACLHS-
: `psaclhs-`: String for the current lock held, from the PSACLHS field of the PSA.

PSACLHSE-
: `psaclhse-`: Extended string for the current lock held, from the PSACLHSE field of the PSA.

PSALOCAL
`psalocal`: Locally locked address space indicator, from the PSALOCAL field of the PSA.

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on the logrec data set records.

RCVY Trace Entries

Purpose

A RCVY trace entry represents entry into a recovery routine following an error or interruption.

Reentry After Certain RCVY Events

Five types of recovery events require reentry in a new environment or address space. See the table below to see when an RCVY trace event requires reentry:

Trace Entry for Recovery Event	Reentry	Trace Entry for Reentry
RCVY ABT	Required only if the task to be ended resides in an address space other than the current home address space	
RCVY ITRM	Always required	RCVY ITRR, if the unit of work ending is locally locked or has an EUT FRR established
RCVY MEM	Always required	RCVY MEMR
RCVY ABTR		
RCVY RCML	Always required	RCVY RCMR
RCVY STRM	Always required	RCVY STRR, if the unit of work ending is in SRB mode, is locally locked, or has an EUT FRR established

Entry Formats

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS-  PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE-

pr home tcb-addr *RCVY ABRT                    trk-----                    psaclhs-  psalocal pasd sasd timestamp-------
                                                                           psaclhse-

pr home tcb-addr *RCVY ABT           return--  comp---- reas---- rc------  psaclhs-  psalocal pasd sasd timestamp-------
                                               asid---- tcb-----           pasclhse-

pr home tcb-addr *RCVY ABTR                    comp---- reas---- rc------  psaclhs-  psalocal pasd sasd timestamp-------
                                               asid---- tcb-----           pasclhse-

pr home tcb-addr *RCVY DAT                     comp---- reas---- psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                                                           pasclhse-

pr home tcb-addr *RCVY FRR  frr-new- psw-----  comp---- reas---- psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                                                 fpw-----  psaclhse-

pr home tcb-addr *RCVY ITRM          return--  comp---- reas----           psaclhs-  psalocal pasd sasd timestamp-------
                                               pppp---- pppp----           psaclhse-

pr home tcb-addr *RCVY ITRR                    comp---- reas----           psaclhs-  psalocal pasd sasd timestamp-------
                                               pppp---- pppp----           psaclhse-

pr home tcb-addr *RCVY MCH                     comp---- reas---- psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                                                           psaclhse-

pr home tcb-addr *RCVY MEM           return--  comp---- reas---- rc------  psaclhs-  psalocal pasd sasd timestamp-------
                                               asid----                    psaclhse-

pr home tcb-addr *RCVY MEMR                    comp---- reas----           psaclhs-  psalocal pasd sasd timestamp-------
                                               asid----                    psaclhse-

pr home tcb-addr *RCVY PERC                    comp---- reas----           psaclhs-  psalocal pasd sasd timestamp-------
                                                                 fpw-----  psaclhse-

pr home tcb-addr *RCVY PROG                    comp---- reas---- psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                                                           psaclhse-

pr home tcb-addr *RCVY RCML          return--  comp---- reas---- asid----  psaclhs-  psalocal pasd sasd timestamp-------
                                               pppp---- pppp----           psaclhse-

pr home tcb-addr *RCVY RCMR                    comp---- reas----           psaclhs-  psalocal pasd sasd timestamp-------
                                               pppp---- pppp----           psaclhse-

pr home tcb-addr *RCVY RESM retry--- psw-----  comp---- reas---- psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                               cpu-----          fpw-----  psaclhse-

pr home tcb-addr *RCVY RSRT                    comp---- reas---- psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                                                           psaclhse-

pr home tcb-addr *RCVY RTRY retry--- psw-----  comp---- reas---- psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                                                 fpw-----  psaclhse-

pr home tcb-addr *RCVY SABN                    comp---- reas---- psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                                                           psaclhse-

pr home tcb-addr *RCVY SPRC                    comp---- reas---- psasuper  psaclhs-  psalocal pasd sasd timestamp-------
                                               asid---- tcb----- fpw-----  psaclhse-

pr home tcb-addr *RCVY STRM          return--  comp---- reas---- tcb-----  psaclhs-  psalocal pasd sasd timestamp-------
                                               pppp---- pppp----           psaclhse-

pr home tcb-addr *RCVY STRR                    comp---- reas---- tcb----   psaclhs-  psalocal pasd sasd timestamp-------
                                               pppp---- pppp----           psaclhse-
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`home`: Home address space identifier (ASID) associated with the TTE.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task or the work element block (WEB).

IDENT
The TTE identifier, as follows:

RCVY Recovery event

An asterisk (*) always appears before RCVY to indicate an unusual condition.

CD/D
Type of recovery event, as follows:
`ABRT`: Abort processing for an unrecoverable error during any recovery termination management (RTM) processing
`ABT`: Request for abnormal end of a task by a CALLRTM TYPE=ABTERM macro, with a system or user completion code
`ABTR`: Rescheduling of a CALLRTM TYPE=ABTERM request for end of a task, when the task is not in the home address space
`DAT`: RTM1 entered for a dynamic address translation (DAT) error
`FRR`: RTM1 processing to invoke a function recovery routine (FRR)
`ITRM`: The system requested RTM1 to end an interrupted task
`ITRR`: ITRM reentry, to process a request to end an interrupted task
`MCH`: RTM1 entered for a machine check interruption
`MEM`: Request for abnormal memory end by a CALLRTM TYPE=MEMTERM macro, with a completion code
`MEMR`: Processing for an abnormal memory end following a MEM event
`PERC`: Percolation from RTM1 to RTM2 to continue recovery processing
`PROG`: RTM1 was entered for a program check interruption
`RCML`: RTM1 was entered to perform special end processing for a task in a failing address space. The failing address space held the local lock of another address space.
`RCMR`: RCML reentry, to process an abnormal end by a resource manager
`RESM`: Resume from an FRR after a RESTART request following an RSRT entry
`RSRT`: RTM entered for a RESTART request from the operator
`RTRY`: Retry from an FRR
`SABN`: The system requested RTM1 to end abnormally the current unit of work
`SPRC`: Final percolation from service request block (SRB) recovery
`STRM`: The system requested RTM1 to end abnormally a suspended task
`STRR`: STRM reentry, to process the abnormal end of a suspended task

PSW----- ADDRESS-
`return--`: Caller's return address
`frr-new- psw-----`: New program status word (PSW) to give control to the FRR
`retry--- psw-----`: Retry PSW

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
`asid----`: Target ASID for end processing.
In a SPRC entry, the ASID is for the task that will be abnormally ended by SRB-to-task percolation. If this field and the

`tcb-----` field are zero, then no SRB-to-task percolation is performed.
`comp----`: System or user completion code
`cpu-----`: Target processor for a restart error indicated on a request for an FRR resume, after an operator RESTART request
`fpw-----`: FRR processing word, in the following format:

```
rsxxxxxp xxxxxxxx ssssssss eeeeeeee
```

r Bit 0 = 1 means a resource manager entry to the FRR

s Bit 1 = 1 means the FRR was skipped

p Bit 7 = 0 means a not serialized SRB-to-task percolation

Bit 7 = 1 means a serialized SRB-to-task percolation

ssssssss
The stack index, which is an index of the FRR stack. The index means the following:
0 Normal stack
1 SVC I/O dispatcher super stack
2 Machine check super stack
3 PC FLIH super stack
4 External FLIH super stack 1
5 External FLIH super stack 2
6 External FLIH super stack 3
7 Restart super stack
8 ACR super stack
9 RTM super stack

eeeeeeee
The entry index, which is an index of the FRR entry on the stack. The index ranges from 0 through 16. If the current stack is a super stack, an index of 0 indicates a super FRR.

`pppp---- pppp----`: PSW of the interrupted unit of work.
The instruction in the PSW may not be the cause of the failure. For example, an interruption can occur because a time limit expired, so that the interrupted instruction is not at fault.

`rc------`: Return code from CALLRTM

`reas----`: Reason code accompanying the completion code appearing in the entry. If not provided, `NONE`.

`psasuper`: PSASUPER field in the prefix save area (PSA)

`tasn----`: Target ASID for RCML reentry

`tcb-----`: Target task control block (TCB) for end processing
In a SPRC entry, the TCB is for the task that will be abnormally ended by SRB-to-task percolation. If this field and the `asid----` field are zero, then no SRB-to-task percolation is performed.
In a STRM or STRR entry, a TCB address of zero indicates that the request was for ending of a suspended SRB.

`trk-----`: RTM1 error tracking area

PSACLHS-
`psaclhs-`: String for the current lock held, from the PSACLHS field of the PSA.

PSACLHSE-
`psaclhse-`: Extended string for the current lock held, from the PSACLHSE field of the PSA.

PSALOCAL
`psalocal`: Locally locked address space indicator, from the PSALOCAL field of the PSA.

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on the logrec data set records.

SVC, SVCE, and SVCR Trace Entries

Purpose

These trace entries represent a supervisor event:
- An SVC trace entry is for processing of a Supervisor Call (SVC) instruction
- An SVCE trace entry is for an error during processing of an SVC instruction
- An SVCR trace entry is for return from SVC instruction processing

Entry Formats

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

pr home tcb-addr  SVC  code svc-old- psw-----  gpr15--- gpr0---- gpr1----                            timestamp-------

pr home tcb-addr  SVCE code svc-old- psw-----  gpr15--- gpr0---- gpr1---- psaclhs-  psalocal pasd sasd timestamp-------
                                                                          psaclhse-

pr home tcb-addr  SVCR code ret-new- psw-----  gpr15--- gpr0---- gpr1----                            timestamp-------
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`home`: Home address space identifier (ASID) associated with the TTE.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task or the work element block (WEB).

IDENT
The TTE identifier, as follows:
SVC Supervisor call (SVC) interruption
SVCE SVC error
SVCR SVC return

An asterisk before SVC, SVCE, or SVCR indicates that the SVC is for an abend (SVC D) and the abend is not for a normal end of task, that is, bit X'08' in the leftmost byte of register 1 (in the UNIQUE-3 column) is not on.

CD/D
`code`: SVC number

PSW----- ADDRESS-
`ret-new- psw`: Program status word (PSW) to receive control when the SVC is dispatched again
`svc-old- psw`: SVC old PSW

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
`gpr15--- gpr0---- gpr1----`: General registers 15, 0, and 1

PSACLHS-
`psaclhs-`: String for the current lock held, from the PSACLHS field of the PSA.

PSACLHSE-
`psaclhse-`: Extended string for the current lock held, from the PSACLHSE field of the PSA.

PSALOCAL
`psalocal`: Locally locked address space indicator, from the PSALOCAL field of the PSA.

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on the logrec data set records.

SSRV Trace Entries

Purpose

An SSRV trace entry represents entry to a system service. The service can be entered by a PC instruction or a branch.

Entry Format

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

pr home tcb-addr  SSRV ssid          return--  data---- data---- data----  psaclhs- psalocal pasd sasd timestamp-------
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`home`: Home address space identifier (ASID) associated with the TTE.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task or the work element block (WEB).

IDENT
The TTE identifier, as follows:

SSRV   Request for a system service

CD/D
`ssid`: SSRV entry identifier

The SSRV entry identifiers are:

ssid (hexadecimal)	Macro for SSRV Request	Component
0001	WAIT	Task management
0002	POST	Task management
0004	GETMAIN	Virtual storage management
0005	FREEMAIN	Virtual storage management
000A	GETMAIN, FREEMAIN	Virtual storage management
005F	SYSEVENT	System resource manager
0078	GETMAIN, FREEMAIN	Virtual storage management
007A	SPI, SPIINT	Service processor interface

ssid (hexadecimal)	Macro for SSRV Request	Component
0100	ETCON	PC/AUTH
0101	ETCRE	PC/AUTH
0102	ATSET	PC/AUTH
0103	AXSET	PC/AUTH
0104	AXEXT	PC/AUTH
0105	AXFRE	PC/AUTH
0106	AXRES	PC/AUTH
0107	ETDES	PC/AUTH
0108	ETDIS	PC/AUTH
0109	LXFRE	PC/AUTH
010A	LXRES	PC/AUTH
010E	SUSPEND	Supervisor control
010F	RESUME	Supervisor control
0110	SCHEDULE	Supervisor control
0111	SCHEDULE	Supervisor control
0112	SCHEDULE	Supervisor control
0113	DSGNL	Supervisor control
0114	RISGNL	Supervisor control
0115	RPSGNL	Supervisor control
0116	SCHEDULE	Supervisor Control
0117	SCHEDULE	Supervisor Control
0118	SUSPEND	Supervisor Control
0119	RESUME	Supervisor Control
011A	RESUME	Supervisor Control
011B	RESUME	Supervisor Control
011C	SCHEDULE	Supervisor Control
011D	IEAMSCHD	Supervisor Control
011E	IEAVPSE or IEAVXFR	Supervisor Control
011F	IEAVRLS or IEAVXFR	Supervisor Control
0128	WAIT	Task management
0129	POST	Task management
012A	POST	Task management
012B	POST	Task management
012C	ASCBCHAP	Task management
012D	STATUS	Task management
012E	STATUS	Task management
0132	STORAGE OBTAIN	Virtual storage management
0133	STORAGE RELEASE	Virtual storage management
0146	SPI, SPIINT	Service processor interface

PSW----- ADDRESS-

`return--`:

- For PC/AUTH, supervisor control, and task management: Caller's return address if the service was entered by a branch; 0 if the service was entered by a PC instruction
- For virtual storage management: The value of the ALET of the storage to be obtained or released

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6

`data----`: Data. The unique trace data for each event is obtained from data areas. The areas for PC/AUTH, supervisor control, and task management are in the *z/OS MVS Data Areas, Vol 5 (SSAG-XTLST)*.

- For an SSRV request to the PC/AUTH component: the PCTRC data area
- For an SSRV request to supervisor control: the SPTRC data area
- For an SSRV request to task management: the TMTRC data area
- For an SSRV request to virtual storage management, the data is:
  - Under UNIQUE-1: Information input to the VSM storage service:
    Byte 0 - Reserved
    Byte 1 (bits 8 through 11) - Storage key
    Byte 2 - Subpool number
    Byte 3 - Request flags:
    `1... ....` ALET operand specified
    `.1.. ....` Storage can be backed anywhere
    `..00 ....` Storage must have caller's residency
    `..01 ....` Storage must have a 24-bit address
    `..10 ....` The request is for an explicit address
    `..11 ....` Storage can have a 24- or 31-bit address
    `.... 1...` Maximum and minimum request
    `.... .1..` Storage should be on a page boundary
    `.... ..1.` Unconditional request
    `.... ...0` OBTAIN request
    `.... ...1` FREEMAIN request
  - Under UNIQUE-2:
    - In an SSRV trace entry for a VSM STORAGE OBTAIN or GETMAIN, one of the following:
      - The length of the storage successfully obtained
      - The minimum storage requested, if the storage was not obtained
    - In an SSRV trace entry for a VSM STORAGE RELEASE or FREEMAIN:
      - the length of the storage to be released, or zero if a subpool release was requested.
  - Under UNIQUE-3:
    - In an SSRV trace entry for a VSM STORAGE OBTAIN or GETMAIN, one of the following:
      - The address of the storage successfully obtained, if you specified address; otherwise, zero.
      - The maximum storage requested, if the storage was not obtained
    - In an SSRV trace entry for a VSM STORAGE RELEASE or FREEMAIN:
      - The address of the storage to be released.
  - Under UNIQUE-4:
    - Left 2 bytes under UNIQUE-4: ASID of the target address space
    - Next byte under UNIQUE-4: Reserved
    - Right byte under UNIQUE-4: Return code from the storage service

PSACLHS-
`psaclhs-`: String for the current lock held, from the PSACLHS field of the PSA.

PSALOCAL
`psalocal`: Locally locked address space indicator, from the PSALOCAL field of the PSA.

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on the logrec data set records.

TIME Trace Entries

Purpose

A TIME trace entry represents a dynamic time-of-day (TOD) clock adjustment by the timer services component.

Entry Formats

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6  PSACLHSE                   TIMESTAMP-RECORD

pr home tcb-addr  TIME  CODE                   word1--- word2--- data----                    pasd sasd timestamp-------
                                               data---- data---- data----
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`home`: Home address space identifier (ASID) associated with the TTE.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task or the work element block (WEB).

IDENT
The TTE identifier, as follows:

TIME   Timer service

CD/D
`code`: Contains a value of 1, indicating that word1 and word2 contain the amount of time that the system advances the time-of-day (TOD) clock when the TOD clock and the External Time Reference (ETR) get out of synchronization.

PSW----- ADDRESS-
`return`: Return address of the program that issued the PTRACE macro

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
`word1, word2`: For a code value of 1, the amount of time that the system advances the TOD clock when the TOD clock and the ETR get out of synch.

PSACLHS-
Blank

PSACLHSE-
Blank

PSALOCAL
Blank

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on the logrec data set records.

USRn Trace Entries

Purpose

A USRn trace entry represents processing of a PTRACE macro in an authorized program. The trace entry contains data from the macro.

Entry Formats

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

pr home tcb-addr  USRn               return--  data---- data---- data----                    pasd sasd timestamp-------
                                               data---- data---- data----

pr home tcb-addr  USRn               return--  idc- rbc data---- data----                    pasd sasd timestamp-------
                                               data---- data---- data----
```

PR
`pr`: Identifier of the processor that produced the TTE.

ASID
`home`: Home address space identifier (ASID) associated with the TTE.

TCB-ADDR
`tcb-addr`: Address of the task control block (TCB) for the current task or the work element block (WEB).

IDENT
The TTE identifier, as follows:

USRn User event. n is a number from X'0' to X'F'.

CD/D
Blank

PSW----- ADDRESS-
`return`: Return address of the program that issued the PTRACE macro

UNIQUE-1/UNIQUE-2/UNIQUE-3
UNIQUE-4/UNIQUE-5/UNIQUE-6
`data----`: User-defined data from the PTRACE macro
`idc-`: PTRACE identification count
`rbc`: Relative byte count

PSACLHS-
Blank

PSALOCAL
Blank

PASD
`cpsd`: Primary ASID (PASID) at trace entry.

SASD
`sasd`: Secondary ASID (SASID) at trace entry.

TIMESTAMP-RECORD
`timestamp-------`: Time-of-day (TOD) clock value when system trace created the trace entry. The value is in the same format as the time stamp on the logrec data set records.

Multiple Trace Entries for a User Event

A single user event appears in more than one trace entry if the PTRACE macro requests recording of more than 5 fullwords of trace data.

For example, the following PTRACE macro requests recording of 11 fullwords of trace data:

```
PTRACE TYPE=USER3,REGS=(2,12),SAVEAREA=STANDARD
```

For this macro, system trace places three entries in the trace table. The entries contain the following:

- The first entry contains the 5 fullwords of trace data in registers 2 through 6.
- The second entry contains the 5 fullwords of trace data in registers 7 through 11.
- The third entry contains the fullword of trace data in register 12.

If the program issuing the PTRACE macro can be interrupted, the three trace entries may not be consecutive in the trace table. The multiple entries contain continuation information as the data for UNIQUE-1. The format of the continuation information is:

```
nnnn hhh
```

nnnn
The PTRACE identification count. This is a hexadecimal number assigned by PTRACE to all entries for one macro processing.

hhh
The byte offset, in hexadecimal, of the next byte of trace data, which is under UNIQUE-2. For the first entry, the offset is X'000'. For the second entry, the offset is X'014'. For the third entry, the offset is X'028'.

The following example shows three trace entries from the preceding PTRACE macro.

```
PR ASID TCB-ADDR  IDENT CD/D PSW----- ADDRESS- UNIQUE-1 UNIQUE-2 UNIQUE-3  PSACLHS- PSALOCAL PASD SASD TIMESTAMP-RECORD
                                               UNIQUE-4 UNIQUE-5 UNIQUE-6

01 000C 00AFF090  USR3               81A007B6  003D 000 00000001 00000002                   000C 000C 9DAA507461CB3E02
                                               00000003 00000004 00000005
01 000C 00AFF090  USR3               81A007B6  003D 014 00000006 00000007                   000C 000C 9DAA507461CB3E02
                                               00000008 00000009 0000000A
01 000C 00AFF090  USR3               81A007B6  003D 028 0000000B                            000C 000C 9DAA507461CB3E02
```

Chapter 9. Master Trace

Master trace is the system's answering machine. It collects your messages, discards the old ones, and you never get those annoying hang-up calls.

Master trace maintains a table of the system messages that are routed to the hardcopy log. This creates a log of external system activity, while the other traces log internal system activity. Master trace is activated automatically at system initialization, but you can turn it on or off using the TRACE command.

Master trace can help you diagnose a problem by providing a log of the most recently issued system messages. For example, master trace output in a dump contains system messages that may be more pertinent to your problem than the usual component messages issued with a dump.

Major Topics

The following topics in this chapter describe master trace:

- "Master Trace and the Hardcopy Log"
- "Customizing Master Trace" on page 9-2
- "Requesting Master Trace" on page 9-2
- "Receiving Master Trace" on page 9-3
- "Reading Master Trace Data" on page 9-4 includes the following topics:
  – "Master Trace Output Formatted in a Dump" on page 9-4
  – "Master Trace Table in Storage" on page 9-5

Master Trace and the Hardcopy Log

Master trace lists the same messages that the system saves automatically and permanently in the hardcopy log, but the entries are maintained in a wraparound table, which means that master trace overwrites old entries when the table is full. You can use master trace data in a dump as a substitute for the hardcopy log when the dump contains the required messages. If the master trace table wraps and overwrites the messages related to your problem before you can request a dump, the dump will not contain useful messages.

In the following example, the dump will contain the required messages:
- The master trace table wraps at 9 pm
- The system issues messages related to a problem between 9:10 and 9:20 pm
- The system issues an SVC dump at 9:30 pm

In this example, the messages pertinent to the problem will be in the master trace data in the dump, since the problem occurred between the time the trace table wrapped and the dump was issued.

To print the system-managed data set containing the hardcopy log, use the JESDS parameter of the OUTPUT JCL statement.

Customizing Master Trace

At initialization, the master scheduler sets up a master trace table of 24 kilobytes. A 24-kilobyte table holds about 336 messages, assuming an average length of 40 characters. You can change the size of the master trace table or specify that no trace table be used by changing the parameters in the SCHEDxx parmlib member.

You can also change the size of the table using the TRACE command. For example, to change the trace table size to 36 kilobytes, enter:

```
TRACE MT,36K
```

Reference

See *z/OS MVS Initialization and Tuning Reference* for the SCHEDxx member.

Requesting Master Trace

Start, change, or stop master tracing by entering a TRACE operator command from a console with master authority. For example, to start the master tracing:

```
TRACE MT
```

To stop master tracing:

```
TRACE MT,OFF
```

You can also use the TRACE command to obtain the current status of the master trace. The system displays the status in message IEE839I. For example, to ask for the status of the trace, enter:

```
TRACE STATUS
```

Example: TRACE STATUS Output

In the following output, master tracing is active with a trace table of 140 kilobytes, as indicated by **MT=(ON,140K)**:

```
TRACE STATUS
IEE839I ST=(ON,0500K,01000K) AS=ON  BR=OFF EX=ON  MT=(ON,140K)
        ISSUE DISPLAY TRACE CMD FOR SYSTEM AND COMPONENT TRACE STATUS
```

If you want to check the current status of system, master, and component tracing, use the DISPLAY TRACE command. The system displays the status in message IEE843I. For example, to ask for the status of the three traces, enter:

```
DISPLAY TRACE
```

Example: DISPLAY TRACE Output

In the following example, master tracing is active with a master trace table of 140 kilobytes, as indicated by MT=(ON,140K):

```
DISPLAY TRACE
IEE843I 15.17.14  TRACE DISPLAY 564
        SYSTEM STATUS INFORMATION
 ST=(ON,0500K,01500K) AS=ON  BR=OFF EX=ON  MT=(ON,140K)
 COMPONENT MODE  COMPONENT MODE  COMPONENT MODE  COMPONENT MODE
 --------------------------------------------------------------
 SYSGRS    ON    SYSSPI    OFF   SYSSMS    OFF   SYSDLF    MIN
 SYSOPS    ON    SYSXCF    ON    SYSLLA    MIN   SYSXES    ON
 SYSAPPC   ON    SYSRSM    ON    SYSAOM    OFF   SYSVLF    MIN
 CTTX      MIN
```

References

- See *z/OS MVS System Commands* for the TRACE and DISPLAY operator commands.
- See *z/OS MVS System Messages, Vol 7 (IEB-IEE)* for IEE839I and IEE843I messages.

Receiving Master Trace

Master trace writes trace data in the master trace table in the address space of the master scheduler. You can obtain master trace data in a stand-alone, SVC, or unformatted dump, if the dump options list includes TRT to request trace data. The following table shows the dumps that contain master trace data:

Dump	Master Trace Data in the Dump?
Stand-alone dump	Default
SVC dump for SDUMP or SDUMPX macro	Default
SVC dump for DUMP operator command	Default
SVC dump for SLIP operator command with ACTION=SVCD, ACTION=STDUMP, ACTION=SYNCSVCD, or ACTION=TRDUMP	Default
Any unformatted dump customized to exclude trace data	Yes, Request SDATA=TRT
ABEND dump to SYSABEND	Not available
ABEND dump to SYSMDUMP	Not available
ABEND dump to SYSUDUMP	Not available
SNAP dump	Not available

Format the master trace data by specifying the IPCS VERBEXIT MTRACE subcommand or using the IPCS Trace Processing selection panel.

References

- See *z/OS MVS IPCS Commands* for the VERBEXIT MTRACE subcommand.
- See *z/OS MVS IPCS User's Guide* for the panel.

Reading Master Trace Data

The following topics in this section show the format of master trace entries:
- "Master Trace Output Formatted in a Dump"
- "Master Trace Table in Storage" on page 9-5

Master Trace Output Formatted in a Dump

The entries in the master trace table are listed in first-in, first-out (FIFO) order, which resembles a hardcopy log. The messages might not be in chronological order because the messages might not have been put into the master trace table in the order the messages were issued.

Example of Formatted Master Trace Output

The following output shows master trace data in a dump formatted by IPCS. The subcommand issued on the IPCS Subcommand Entry panel was:

```
VERBEXIT MTRACE
```

```
 *** MASTER TRACE TABLE ***


 TAG  IMM DATA  |------------------------ MESSAGE DATA ------------------>|

 0001 008C7DD4  NR0000000 SP21     87153 17:59:11.09 MASTER   00000000
                IEE794I 865      PENDING OFFLINE
 0001 008AD6A8  N 0000000 SP21     87153 17:59:11.88 STC  877 00000008
                IEF285I   CATALOG.D83LOG                               KEPT
                    10
 0001 008AD6A8  N 0000000 SP21     87153 17:59:11.88 STC  877 00000008
                IEF285I   VOL SER NOS= D83LOG.

 0001 008AD6A8  N 0000000 SP21     87153 17:59:11.99          00000000
                IEA989I SLIP TRAP ID=X33E MATCHED
 0001 008C7EF8  N 1000000 SP21     87153 17:59:12.22 JOB  801 00000000
                IEF234E D 2C3,338003,PUB
 0001 008B51D8  N 1000000 SP21     87153 17:59:12.40 JOB  801 00000000
                IEF281I 2C3 NOW OFFLINE
 0001 008B52FC  NR0000000 SP21     87153 17:59:12.41 MASTER   00000000
                IEE794I 2C6      PENDING OFFLINE
 0001 008B52FC  N 1000000 SP21     87153 17:59:13.53 JOB  801 00000000
                IEF234E D 2C4,338004,STR
 0001 008B51D8  N 1000000 SP21     87153 17:59:14.21 JOB  801 00000000
                IEF281I 2C4 NOW OFFLINE
 0001 008C7EF8  NR0000000 SP21     87153 17:59:14.21 MASTER   00000000
                IEE794I 2E6      PENDING OFFLINE
 0001 008C7DD4  M 0002000 SP21     87153 17:59:14.29 STC  828 00000000 *0002
                System  SYSIEFSD Q4<-----------------------------------
 0001 008B4D48  D                                         372 00000000
                Run  Exc  N1OFF2C0  SP21
 0001 008B4DA8  D                                         372 00000000
                Wait Shr  *MASTER*  SP21         0:41:22
 0001 008B39E4  D                                         372 00000000  0034
                System  SYSIEFSD VARYDEV<------------------------------
```

The meaning of the highlighted text in the preceding example is as follows:

TAG

A halfword containing the identity of the caller. TAG can be one of the following:

Tag	Caller
000	Reserved

001 Communications Task (COMMTASK)

002 Master scheduler

003 Trace command

Current identifiers are defined in the macro, IEZMTPRM, which maps the parameter list.

IMM DATA
A fullword of immediate data, consisting of the 32 bits defined by the caller. The significance of the immediate data is defined by the caller.

MESSAGE DATA
The message. If a problem occurs during processing, the line following the message indicates the problem.

Master Trace Table in Storage

This topic describes master trace data as it is recorded in the master trace table in the master scheduler address space. You can use this information to write your own formatting or analysis routines for master trace information.

Master trace places entries in FIFO order. Thus, a current entry is in front of the older entries. When the table is full, master trace wraps, and resumes recording entries at the end of the table.

Note that the messages may not be in chronological order because the messages may not be put in the master trace table in the order in which they are issued.

Location

Locate the master trace table from the communication vector table (CVT) as follows:

At the following location:	In a field named:	Find the following address:
CVT+X'94'	CVTMSER	IEEBASEA (master scheduler resident data area)
IEEBASEA+ X'8C'	BAMTTBL	Start of the master trace table

Format

The unformatted master trace table in the master scheduler address space contains a header and, for each message logged in the table, an entry. The following two topics show the fields in the header and an entry.

Header in the Master Trace Table

TABLE ID	CURRENT	START	END
SUBPOOL	LENGTH	WRAP TIME	
WRAP	PROCFLAG	DATA LENGTH	RESERVED1
WRKMB808			
RESERVED2			

TABLE ID
A fullword field containing *MTT*. MTT is an eye-catcher to mark the beginning of the master trace table.

CURRENT
A fullword field containing the address of the current (most recently stored) entry.

START
A fullword field containing the address of the first byte of the trace area.

END
A fullword field containing the address of the first byte beyond the end of the trace area.

SUBPOOL
A fullword field containing the number of the subpool in which this table resides.

LENGTH
A fullword field containing the length, in bytes, of the table header and the area containing the entries. This length is the default table size or the size specified on the TRACE command.

WRAP TIME
A double word field containing a time, either when the table was initialized or when the last table wrap occurred. The time is in the form:

HH	hours
MM	minutes
SS	seconds
T	tenths of a second

WRAP
A fullword field containing the address of the first byte of the last entry stored before the most recent table wrap.

Note: This address is initialized to zero and remains zero until the first table wrap.

PROCFLAG
Internal processing flags used by master trace.

DATA LENGTH
A fullword field containing the length, in bytes, of the data area part of the table.

RESERVED1
A fullword field.

WRKMB808
A 16-word work area for use by master trace and serialized by the CMS or LOCAL locks.

RESERVED2
A 4-word field.

Entry in the Master Trace Table

Entry header				CALLER-PASSED DATA
FLAGS	TAG	IMM DATA	LEN	

Entry header
10-byte header for the entry.

FLAGS
A halfword containing the flags set by the caller in the parameter list passed to master trace.

TAG
A halfword containing the identity of the caller. TAG can be one of the following:

Tag	Caller
0000	Reserved
0001	Communications Task (COMMTASK)
0002	Master scheduler
0003	Trace command

Current identifiers are defined in the macro, IEZMTPRM, which maps the parameter list.

IMM DATA
A fullword containing 32 bits defined by the caller. Master trace stores these bits in the table without checking them for validity.

The significance of IMMEDIATE DATA is defined by the caller; likely values are a counter, a control block address, or flags describing the passed data.

LEN
A halfword containing the length of the caller-passed data.

CALLER-PASSED DATA
A variable-length field containing the data provided by the caller.

The master trace table entries vary in length. If the caller specifies the length of the caller-passed data as zero, the entry in the master trace table consists of only the 10-byte header.

Chapter 10. The Generalized Trace Facility (GTF)

Using GTF is like programming your VCR to tape that important television show while you're away from home. It takes some time and planning to set it up right, but you get to see your show!

The generalized trace facility (GTF) is a service aid you can use to record and diagnose system and program problems. GTF is part of the MVS system product, and you must explicitly activate it by entering a START GTF command.

Using GTF, you can record a variety of system and program events on all of the processors in your installation. If you use the IBM-supplied defaults, GTF lists many of the events that system trace lists, showing minimal data about them. However, because GTF uses more resources and processor time than system trace, IBM recommends that you use GTF when you experience a problem, selecting one or two events that you think might point to the source of your problem. This will give you detailed information that can help you diagnose the problem. You can trace combinations of events, specific incidences of one type of event, or user-defined program events that the GTRACE macro generates. For example, you can trace:

- Channel programs and associated data for start and resume subchannel operations, in combination with I/O interruptions
- I/O interruptions on one particular device
- System recovery routine operations

The events that GTF traces are specified as options in a parmlib member. You can use the IBM supplied parmlib member or provide your own. Details of GTF operation, which include storage needed, where output goes and recovery for GTF, are defined in a cataloged procedure in SYS1.PROCLIB.

GTF can trace system and program events both above and below 16 megabytes. For each event it traces, GTF produces trace records as its output. You can have GTF direct this output to one of the following places:

- A trace table in virtual storage.
- A data set on a tape or direct access storage device (DASD).

Choose a trace table for your GTF output when maintaining good system performance is very important to your installation. The trace table cannot contain as much GTF trace data as a data set, but will not impact performance as much as a data set because there is no I/O overhead.

Choose a data set or sets when you want to collect more data than will fit in a trace table. Writing trace data to a data set does involve I/O overhead, so choosing this option will impact system performance more than a trace table.

GTF can use only one table in virtual storage, but can use up to 16 data sets. If you specify more than one data set, all of them must reside on devices of the same class, tape, or DASD.

The following briefly highlights the relationship between GTF and other diagnostic tools and services:

GTF and IPCS

You can use IPCS to merge, format, display, and print GTF output.

References

See *z/OS MVS IPCS User's Guide* and *z/OS MVS IPCS Commands* for information about the COPYTRC, GTFTRACE, and MERGE subcommands, and the trace processing option of the IPCS dialog.

GTF and the GTRACE Macro

You can use GTF in combination with the GTRACE macro, provided you activate GTF with TRACE=USR. Then, your programs can issue GTRACE macros to generate trace records, which GTF can store in the trace table.

Reference

See *z/OS MVS Programming: Authorized Assembler Services Reference ENF-IXG* for information about coding the GTRACE macro.

GTF and System Trace

You can use GTF in combination with system trace. System trace records predetermined system events, and provides minimal details about each event. Supplement system trace information by selecting specific GTF options to provide more detailed information about system and user events. For further information about system trace, see Chapter 8, "System Trace" on page 8-1.

GTF and Indexed VTOC Processing

You can use GTF to trace events that occur during processing for indexed volume table of contents (VTOC).

Reference

See *z/OS DFSMSdfp Diagnosis Reference* for more information.

The following topics in this chapter describe GTF in detail:

- "Using IBM Defaults for GTF" on page 10-3 describes the IBM-supplied. parmlib member (GTFPARM) and the IBM-supplied cataloged procedure.
- "Customizing GTF" on page 10-4 describes how to customize GTF by either overriding IBM's defaults or providing your own parmlib member and cataloged procedure.
- "Starting GTF" on page 10-10.
- "Stopping GTF" on page 10-16.
- "GTF Trace Options" on page 10-18 describes options you can use to select events for GTF tracing. This topic also has guidelines for combining GTF options and specifying keywords for prompting options.
- "Receiving GTF Traces" on page 10-31 describes how to request a dump containing GTF trace data and how to format the GTF data in dumps or data sets using IPCS. This topic also describes how to consolidate multiple sources of GTF trace data, how to extract GTF trace data from dumps, and how to merge GTF trace data into chronological sequence.
- "Reading GTF Output" on page 10-34 describes the format of the GTF trace records, both formatted and unformatted.

Using IBM Defaults for GTF

IBM supplies both a SYS1.PARMLIB (also called parmlib) member that contains predefined GTF trace options and a cataloged procedure for GTF, should you want to use IBM's defaults for GTF operation. You can override some of the default options by specifying certain parameters on the START command that activates GTF.

The IBM-Supplied Parmlib Member of GTF Trace Options

IBM supplies the GTFPARM parmlib member, which contains the GTF trace options shown below:

```
TRACE=SYSM,USR,TRC,DSP,PCI,SRM
```

Briefly, these options request that GTF trace the following:

SYSM Selected system events

USR User data that the GTRACE macro passes to GTF

TRC Trace events associated with GTF itself

DSP Dispatchable units of work

PCI Program-controlled I/O interruptions

SRM Trace data associated with the system resource manager (SRM)

For complete descriptions of these trace options, see "GTF Trace Options" on page 10-18.

The IBM-Supplied Cataloged Procedure

IBM supplies the GTF cataloged procedure, which resides in SYS1.PROCLIB. The cataloged procedure defines GTF operation, including storage needed, where output is to go, recovery for GTF, trace output data sets, etc. Figure 10-1 illustrates the content of the IBM supplied cataloged procedure.

```
//GTFNEW  PROC MEMBER=GTFPARM
//IEFPROC EXEC PGM=AHLGTF,PARM='MODE=EXT,DEBUG=NO,TIME=YES',
//  TIME=1440,REGION=2880K
//IEFRDER DD   DSNAME=SYS1.TRACE,UNIT=SYSDA,SPACE=(TRK,20),
//             DISP=(NEW,KEEP)
//SYSLIB  DD   DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
```

Figure 10-1. IBM-Supplied GTF Cataloged Procedure

The statements in this cataloged procedure are:

PROC
Defines the GTF cataloged procedure.

EXEC

PGM=AHLGTF
Calls for the system to run program AHLGTF.

PARM='MODE=EXT,DEBUG=NO,TIME=YES',
The parameters selected specify that GTF direct trace data to a data set on tape or DASD, attempt recovery if it encounters an error, and give every

trace record a time stamp. See the explanation for the EXEC statement under “Setting Up a Cataloged Procedure” for detailed information.

TIME=1440
The amount of time, in seconds, that GTF will remain active.

REGION=2880K
Specifies the maximum size of the storage that GTF requires.

IEFRDER DD
Defines the trace output data set, according to the following defaults:

- The trace output data set has the name SYS1.TRACE
- The data set resides on a DASD that is large enough for the data set to contain 20 physical blocks. After completely filling the 20 physical blocks, GTF will overlay previously written records with new trace records, starting at the beginning of the output data set.

SYSLIB DD
Defines the IBM-supplied GTFPARM parmlib member that contains GTF trace options and their default values.

Customizing GTF

You can customize GTF to the needs of your installation by either overriding IBM’s defaults through the START GTF command, or providing your own parmlib member and cataloged procedure for GTF.

Customize GTF in one of the following ways:

- Predefine the GTF trace options in a parmlib or data set member. See “Defining GTF Trace Options”.
- Set up a cataloged procedure. See “Setting Up a Cataloged Procedure”.
- Override the defaults in the IBM supplied GTF cataloged procedure using the START command. See “Using the START Command to Invoke GTF” on page 10-11.
- Have the operator specify trace options directly through the console after entering the START command. See “Specifying or Changing GTF Trace Options through System Prompting” on page 10-12.

This section also includes information about how to determine how much storage GTF needs for the trace options you choose. See “Determining GTF’s Storage Requirements” on page 10-9.

Defining GTF Trace Options

If you supply your own parmlib member or data set containing GTF trace options, you can select any of the options listed in “GTF Trace Options” on page 10-18.

The member containing predefined trace options does not have to reside in the parmlib member. GTF will accept any data set specified in the SYSLIB DD statement of the cataloged procedure, or in the START command, as long as that data set’s attributes are compatible with those of SYS1.PARMLIB.

Setting Up a Cataloged Procedure

Set up your own GTF cataloged procedure when you want to control details of GTF operation such as:

- Amount of storage needed for tracing

- Recovery for GTF
- Number and type of trace output data sets.

If you choose to supply your own cataloged procedure, include the following statements:

PROC
Defines your cataloged procedure.

EXEC

PGM=AHLGTF
Calls for the system to run program AHLGTF.

PARM='*parm, parm***...'**
Options specified on the PARM parameter specify where GTF writes trace data and the amount of storage needed for GTF to collect and save trace data in various dump types. **parm** can be any of the following:

```
MODE={INT|EXT|DEFER}
SADMP={nnnnnnK|nnnnnnM|40K}
SDUMP={nnnnnnK|nnnnnnM|40K}
NOPROMPT
ABDUMP={nnnnnnK|nnnnnnM|0K}
BLOK={nnnnn|nnnnnK|nnnnnM|40K}
TIME=YES
DEBUG={YES|NO}
```

MODE={INT|EXT|DEFER}
Defines where GTF writes the trace data. MODE=INT directs data to a trace table in virtual storage and MODE=EXT directs data to a data set on tape or DASD.

If MODE=INT, GTF will direct the trace data to a trace table in virtual storage, and will ignore any DD statements that define GTF output data sets. Choose this option when it is very important to you to maintain good system performance while GTF runs. The trace table cannot contain as much GTF trace data as a data set, but will not impact performance as much as a data set because there is no I/O overhead.

If MODE=EXT or MODE=DEFER, GTF directs the output to a trace data set defined by GTFOUTxx or IEFRDER DD statements. MODE=EXT is the default value. Choose MODE=EXT or MODE=DEFER when you want to collect more data than will fit in a trace table. Writing trace data to a data set does involve I/O overhead, so choosing one of these options will impact system performance more than MODE=INT.

MODE=DEFER will place the trace data in the GTF address space until the operator enters the STOP GTF command. During GTF end processing, GTF will transfer the data from its address space to the output data set.

The amount of data transferred for MODE=EXT or MODE=DEFER is one of the following:
- The default amount
- The amount specified on the SADMP|SA keyword

When the trace output data set is full, GTF continues as follows:

- **Direct access**: GTF resumes recording at the beginning of the data set, when the primary allocation is full. Thus, GTF writes over earlier trace data.

- **Tape**: GTF writes an end-of-file record. The tape is rewound and unloaded, then a new volume is mounted on the tape unit. If GTF has only one tape data set and only one unit for the data set, GTF does not write trace records while the tape is unavailable, thus losing trace data.

  GTF can write to multiple tape units in two ways:

  – Multiple GTFOUTxx DD statements can specify tape data sets. When GTF fills one data set, it changes to the next data set.

  – The IEFRDER DD statement can specify two tape units; in this case, GTF resumes writing the current trace data on the other unit, while rewinding and unloading the full volume.

SADMP|SA=nnnnnnK|nnnnM|40K}
Specifies the amount of storage needed to save GTF trace data for stand-alone dumps. Specify the amount of storage in terms of either kilobytes (K) or megabytes (M). The minimum amount is 40K, and the maximum is 2048M minus 400K, or 2096752K. GTF rounds up the amount to the block size boundaries for DASD data sets, or 32K boundaries for tape data sets or internal mode. The default value for this parameter is 40K (rounded up to the correct boundary).

SDUMP|SD={nnnnnnK|nnnnM|40K}
Specifies the amount of storage needed to save GTF trace data for SVC dumps. Specify the amount of storage in terms of either kilobytes (K) or megabytes (M). The minimum amount is zero, and the maximum cannot exceed the maximum amount of storage defined by the SADMP parameter. GTF rounds up the amount to the block size boundaries for DASD data sets, or 32K boundaries for tape data sets or internal mode. The default value for this parameter is 40K (rounded up to the correct boundary).

NOPROMPT|NP
If specified, indicates that the operator will not be prompted to specify trace options. Message AHL125A and AHL100A will not be issued. Use this parameter when you have a parmlib member set up with the desired GTF options and you want to avoid multiple replies in a sysplex environment.

ABDUMP|AB={nnnnnnK|nnnnM|0K}
Specifies the amount of GTF trace data to be formatted in an ABEND or SNAP dump. Specify the amount of trace data in terms of either kilobytes (K) or megabytes (M). The minimum amount is zero, and the maximum cannot exceed the maximum amount of storage defined by the SADMP parameter. GTF rounds up the amount to the block size boundaries for DASD data sets, or 64K boundaries for tape data sets or internal mode. The default value for this parameter is 0K, which means that no GTF data will appear in ABEND or SNAP dumps.

For ABEND or SNAP dumps. GTF formats only those records that are directly associated with the failing address space. GTF does not format the channel program trace data associated with the failing address space.

BLOK={nnnnn|nnnnnK|nnnnnM|40K}
Specifies the amount of virtual storage (E)CSA that GTF will use to collect trace data. Specify this storage amount in 4096-byte pages

(nnnnn), or in bytes (nnnnnK or nnnnnM). The maximum amount is 99999; the default is 40K. If the amount is less than 40K, GTF will use the default.

TIME=YES
Specifies that every GTF trace record have a time stamp, as well as the block time stamp associated with every block of data. The time stamp is the eight-byte time of day (TOD) clock value at the local time in which GTF puts the record into the trace table. GTF does not accept TIME=NO; all output records will have time stamps.

When you use IPCS to format and print the trace records, a time stamp record follows each trace record. You can use these time stamp records to calculate the elapsed time between trace entries. The time stamp record is described in "Time Stamp Records" on page 10-39.

DEBUG={YES|NO}
Specifies whether GTF should attempt recovery after it encounters an error. If DEBUG=YES, GTF will not attempt any recovery. Instead, GTF will issue an error message and end after encountering any error, so that the contents of the trace table immediately prior to the error remain intact.

If DEBUG=NO, which is the default, GTF does the following:

- For errors in GTF processing, GTF continues processing after doing one or more of the following:
  - Flagging the trace record or trace record field associated with the error
  - Issuing a message to the console to notify the operator that an error occurred
  - Suppressing the error or function in which the error occurred.
- For errors that do not occur in GTF itself, GTF ends abnormally. If GTF stops processing, that will not cause any other task to also stop.

TIME=nnnn
Specifies unlimited processor time for GTF.

REGION=nnnnK
Specifies the maximum size of the storage that GTF requires. Specify any value from 832K to 2880K. See "Determining GTF's Storage Requirements" on page 10-9 for information about determining the value for REGION.

IEFRDER DD or GTFOUTxx DD
Defines the trace output data set or data sets. This statement is required only if you do one of the following:
- Specify MODE=EXT or MODE=DEFER
- Use the default MODE=EXT

IEFRDER DD can be used, but does not have to be used, for one trace output data set. Additional data sets must be defined on GTFOUTxx DD statements, where xx is one or two characters that are valid in DDNAMES. See "Guidelines for Defining GTF Trace Output Data Sets in a Cataloged Procedure" on page 10-8 for guidance on how to define output data sets for GTF.

SYSLIB DD
Optionally include a SYSLIB DD statement to define the IBM-supplied member, or the installation-supplied member, that contains GTF trace options. If the member exists, GTF will use the options in that member. If the member does not exist, GTF will issue an error message and stop.

If you code a procedure that does not contain a SYSLIB DD statement, GTF issues message AHL100A to prompt the operator for options after the operator enters the START GTF command. In response, the operator can supply the desired trace options through the console. See "Specifying or Changing GTF Trace Options through System Prompting" on page 10-12 for examples of specifying options through the console.

Guidelines for Defining GTF Trace Output Data Sets in a Cataloged Procedure

Use the following guidelines for specifying trace output data sets on the IEFRDER DD or GTFOUTxx DD statements:

- You can define up to 16 output data sets for GTF to use. If you define more than 16 data sets, GTF will accept the first 16 and ignore the rest.
- If GTF cannot open all of the data sets, it issues a message that identifies those that are unopened, and continues processing with those that are open.
- Do not specify the RLSE option when the SPACE parameter is used because the output data sets will be opened and closed more than once while GTF runs.
- Do not request secondary extents for trace data sets. GTF will only use the first extent.

  To obtain the maximum degree of control over the number of trace entries for which space will be allocated, specify space allocation in blocks, using a block length that matches the BLKSIZE of your trace data set. Do not specify any secondary space. Use the CONTIG option to request contiguous blocks. For example, if your BLKSIZE is 8192, code the SPACE keyword as follows:

  ```
  SPACE=(8192,(500,0),,CONTIG)
  ```
- All data sets must be in the same device class: either DASD or tape, but not both. If you mix device classes, GTF will ignore the tapes and use only DASD. However, the data sets can have different device types; for example, you can mix 3350 and 3380 device types.
- To ensure the most efficient GTF processing, do not specify any particular block size for the output data set or data sets in either:
  - The cataloged procedure for GTF
  - The JCL, TSO/E commands, or interactive system productivity facility/program development facility (ISPF/PDF) panels that you might use to preallocate the data set or data sets

  The system computes an optimal block size when it opens each data set. **EXCEPTION:** If you want GTF to use an unlabeled tape as the output data set, you must specify the logical record length and block size when you allocate that data set.
- If you define more than one data set, you should ensure that the number of paths to the data sets equals the number of data sets.
- You can specify the number of channel programs for each output data set using the NCP parameter on each DD statement. The NCP value determines the rate at which GTF transfers data to the output data sets. For example, if you want to transfer data to your data sets at a rate of 25 buffers per second and you have 5 data sets, you will need to specify an NCP value of 5. GTF then transfers data to the 5 data sets at a rate of 5 buffers per second per data set for a total rate of 25 buffers per second.

  The maximum value for NCP is 255. If you do not specify a value for NCP, or if you specify a value less than four, GTF will use the following default values:
  - For tape: four
  - For DASD: the number of output blocks per track, multiplied by four.

- If, when you enter the START command, you override any of the DD statements for multiple output data sets, you must use symbolic parameters in those DD statements. See “Using the START Command to Invoke GTF” on page 10-11 for more information.

Determining GTF’s Storage Requirements

The storage that GTF requires depends on the trace options you choose. After you have decided which options you want GTF to use, use Figure 10-2 on page 10-10 to determine the amount of storage you should specify in the REGION parameter of either your cataloged procedure’s EXEC statement or the START GTF command. There are several types of storage to calculate:

- Extended pageable link pack area (EPLPA)
- System queue area (SQA)
- Extended common service area (E)CSA
- Region storage

Use the formulas in Figure 10-2 to calculate the amount of storage needed for each storage type. Then add them all together to arrive at the final figure to specify on the REGION parameter. For information about the options mentioned in the figure, see “GTF Trace Options” on page 10-18.

<table>
<tr><th>Extended Pageable Link Pack Area</th><th>System Queue Area</th></tr>
<tr><td>

Fix = Opt + Prmpt + 8K

Fix: Fixed storage in pageable EPLPA while GTF is active.

Opt: Sum of storage required for each GTF option specified. See the table below to calculate OPT.

Prmpt: Optional additional 1.5K if any prompting options specified.

8K: 8K required for services.

</td><td>

SQA = 16500 + REG + SAVE + CBLOC

SQA: System Queue Area storage requirement.

REG: 232 bytes per processor are required for register save areas, regardless of whether or not GTF is active.

SAVE: 1352 bytes per processor are required for save/work areas when GTF is active.

CBLOC: 1700-2200 bytes are needed for control blocks when GTF is active.

Notes:

1. When you specify PCI and either CCW or CCWP, GTF requires the following additional SQA storage:

 16 + 1200 * (value of PCITAB in bytes)

2. When you specify either CCW or CCWP, GTF uses 4096 additional bytes of the SQA for each processor.

3. When you specify USRP, GTF uses 4096 additional bytes ofthe SQA for each processor.

</td></tr>
</table>

Option	Size Required
SYSM	4K
SYS with DSP and/or SRM and/or RNIO	7K
SYS, SYSP	18K
PI, DSP, PIP	2.5K
EXT	2K
IO, IOP, SIO, SIOP, SSCH, SSCHP	6K
SVC, SVCP	10K
SRM, RR, RNIO	3K
SLIP	8K
USR, USRP	1.5K
PCI, TRC	No Requirement
CCW, CCWP	9.3K

Notes:

1. When you specify more than one event from a line, the size requirement is the same as if you specified only one option. For example, DSP and PI require 2.5K.

2. For the maximum storage requirement round up the storage requirement for each option you specified to the nearest 4K boundary.

3. For the minimum storage requirement, round up the 'FIX' value to the nearest 4K boundary.

Example:
1) Options = PIP, DSP, SLIP
 Fix = 10.5 + 1.5 + 8 = 20K minimum or
 = 12 + 1.5 + 8 = 21.5 = 24K maximum
2) Options = SYSM, SRM USR, TRC
 Fix = 8.5 + 0 + 8K = 16.5 = 20K minimum or
 = 12 + 0 + 8K = 20K maximum

Extended Common Service Area (CSA)

ESQA = N

N: Approximately 4500 times the number of blocks specified on the BLOCK = keyword parameter of the GTF START command.

The default is 45056 bytes.

Region Storage

SUBPOOL: GTF uses a default of 1031 kilobytes of storage for true data.

REGION: GTF requires a minimum of an 800K virtual region to run.

Figure 10-2. GTF Storage Requirements

Starting GTF

When you activate GTF, it operates as a system task, in its own address space. The only way to activate GTF is to enter a START GTF command from a console with master authority. Using this command, the operator selects either IBM's or your cataloged procedure for GTF. The cataloged procedure defines GTF operation; you can accept the defaults that the procedure establishes, or change the defaults by having the operator specify certain parameters on the START GTF command.

Because GTF sends messages to a console with master authority, enter the command only on a console that is eligible to be a console with master authority. Otherwise, you cannot view the messages from GTF that verify trace options and other operating information.

Using the START Command to Invoke GTF

The START command, without any parameters other than the IBM-supplied procedure name and an identifier, uses the defaults of the cataloged procedure. If that source JCL contains a DD statement for the data set member of predefined trace options, GTF will issue a message that lists those options, and will allow the operator to override them. Otherwise, GTF will prompt the operator to specify trace options directly through the console. See “Specifying or Changing GTF Trace Options through System Prompting” on page 10-12 for more information.

To invoke GTF, the operator enters the START command shown below:

```
{START|S}{GTF|membername}[.identifier]
```

Reference

For information about this START GTF command and parameters you can use to change the GTF cataloged procedure, see *z/OS MVS System Commands*.

Guidelines for Overriding JCL Statements in the GTF Cataloged Procedure

You can override the parameters of only one output data set using the **keyword=option** parameter on the START command. If you have defined more than one output data set, and you used IEFRDER as the DDNAME for one of the DD statements, the keywords the operator specifies on the START command will override the attributes of the data set that IEFRDER defines. If you want to alter the attributes of another data set, or more than one data set, you must:

- Use symbolic parameters in the JCL DD statements for those attributes you want to change. You cannot use DD statement keywords as symbolic parameters; for example, you cannot code UNIT=&UNIT;
- Assign values to the symbolic parameters in the EXEC or PROC statements in the cataloged procedure.
- Instruct the operator to specify keywords in the START command to override the symbolic parameter values specified on the EXEC or PROC statements.

Examples of Overriding the JCL Statements in the GTF Cataloged Procedure

The following shows examples of setting up a cataloged procedure when you want to override JCL statements in the procedure using the **keyword=option** parameter on the START command. Please note that the DD statement parameters in both of the following procedures are for example only; the needs of your installation might require you to provide DD parameters in addition to, or other than, DSNAME, UNIT, and DISP.

Example: Altering One Data Set

If you want to alter just one data set using the START command, your cataloged procedure could look like the following:

```
//GTFABC    PROC   MEMBER=GTFPARM
//IEFPROC   EXEC   PGM=AHLGTF,REGION=2880K,TIME=1440,
//                 PARM=('MODE=EXT,DEBUG=NO')
//IEFRDER   DD     DSNAME=SYS1.GTFTRC,UNIT=SYSDA,
//                 SPACE=(4096,20),DISP=(NEW,KEEP)
//SYSLIB    DD     DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
//GTFOUT1   DD     DSNAME=SYS1.TRACE1,UNIT=SYSDA,DISP=(NEW,KEEP)
//GTFOUT2   DD     DSNAME=SYS1.TRACE2,UNIT=SYSDA,DISP=(NEW,KEEP)
//GTFOUT3   DD     DSNAME=SYS1.TRACE3,UNIT=SYSDA,DISP=(NEW,KEEP)
```

The operator then enters **START GTFABC,,,UNIT=TAPE**, to alter only the data set that IEFRDER defines.

Example: Altering More Than One Data Set

If you want to alter the attributes of more than one data set with the START command, use the following JCL statements in your cataloged procedure:

```
//GTFABC  PROC MEMBER=GTFPARM,NAME1='SYS1.TRACE1',
//             NAME2='SYS1.TRACE2',NAME3='SYS1.TRACE3',
//IEFPROC EXEC PGM=AHLGTF,REGION=2880K,TIME=1440,
//             DEVICE='SYSDA',DSPS='OLD'
//          PARM=('MODE=EXT,DEBUG=NO')
//SYSLIB  DD   DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
//GTFOUT1 DD   DSNAME=&NAME1,UNIT=&DEVICE,DISP=&DSPS;
//GTFOUT2 DD   DSNAME=&NAME2,UNIT=&DEVICE,DISP=&DSPS;
//GTFOUT3 DD   DSNAME=&NAME3,UNIT=&DEVICE,DISP=&DSPS;
```

The operator then enters **START GTFABC,,,DEVICE=TAPE**, to override the default value of the UNIT parameter for each output data set in your cataloged procedure.

Reference

See *z/OS MVS JCL Reference* for more information about using symbolic parameters in JCL statements.

Specifying or Changing GTF Trace Options through System Prompting

After the operator enters the START command, GTF issues message AHL100A or AHL125A to allow the operator either to specify or to change trace options. If the cataloged procedure or START command did not contain a member of predefined options, GTF issues message AHL100A so the operator may enter the trace options you want GTF to use. If the procedure or command did include a member of predefined options, GTF identifies those options by issuing the console messages AHL121I and AHL103I. Then you can either accept these options, or reject them and have the operator specify new options. This sequence of messages appears as:

```
AHL121I TRACE OPTION INPUT INDICATED FROM MEMBER memname OF PDS dsname
AHL103I TRACE OPTIONS SELECTED -
keywd=(value),...,keywd=(value)
keywd,keywd,...,keywd
AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
```

Note: If you specify NOPROMPT or NP on the START GTF command, the system will not issue message AHL125A to request the respecification of trace options or the continuation of initialization.

If you choose to reject any options in the member, you are rejecting all of the options specified in that member. Instructing the operator to respecify trace options does not modify the options in the data set member.

The format of the response is:

```
TRACE=trace option[,trace option]...
```

Note that the trace options determine the amount of storage GTF requires. See “Determining GTF’s Storage Requirements” on page 10-9.

GTF will accept the trace options listed under “GTF Trace Options” on page 10-18.

Examples of Starting GTF

In this topic you will find the following examples:

- “Starting GTF with a Cataloged Procedure and Parmlib Member”
- “Starting GTF with Internal Tracking” on page 10-14
- “Starting GTF with Trace Output to an Existing Data Set on Tape” on page 10-14
- “Starting GTF with Trace Options Stored in SYS1.PARMLIB” on page 10-14
- “Starting GTF Without Trace Options in a Member” on page 10-15
- “Starting GTF to Trace VTAM Remote Network Activity” on page 10-16

Starting GTF with a Cataloged Procedure and Parmlib Member

This example shows GTF started with a cataloged procedure that indicates the GTFPARM parmlib member. The trace options are specified in the parmlib member record. In this example, message AHL103I displays the options specified in the GTFPARM member: TRACE=SYSM, DSP, PCI, SRM, TRC, USR. This example shows the messages and the reply generated by the initial START command, and that the GTFPARM specifications will be in effect.

Example: Starting GTF with a Cataloged Procedure

```
START GTF.EXAMPLE1

AHL121I TRACE OPTION INPUT INDICATED FROM MEMBER GTFPARM OF PDS SYS1.PARMLIB

AHL103I TRACE OPTIONS SELECTED--SYSM,USR,TRC,DSP,PCI,SRM

00 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

REPLY 00,U

AHL031I GTF INITIALIZATION COMPLETE
```

Starting GTF with Internal Tracking

This example shows GTF started with MODE=INT. The trace data is maintained in virtual storage and is not recorded on an external device. In this example, the operator overrides the trace options given in the supplied parmlib member:

Example: Starting GTF with Internal Tracking

```
START GTF.EXAMPLE2,,,(MODE=INT),DSN=NULLFILE

AHL121I TRACE OPTION INPUT INDICATED FROM MEMBER memname OF PDS dsname

AHL103I TRACE OPTIONS SELECTED - SYSM,USR,TRC,DSP,PCI,SRM

00 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

REPLY 00,TRACE=IO,SSCH,SVC,DSP

AHL103I TRACE OPTIONS SELECTED -- DSP,SVC,IO,SSCH

01 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

REPLY 01,U

AHL031I GTF INITIALIZATION COMPLETE
```

Starting GTF with Trace Output to an Existing Data Set on Tape

This example shows how the START command is used to direct GTF trace output to an existing data set on tape rather than to an existing data set on a DASD. The device type and volume serial number are supplied. The disposition and name of the trace data set are changed from DISP=(NEW,KEEP) and DSNAME=SYS1.TRACE to DISP=(OLD,KEEP) and DSNAME=TPOUTPUT. The specified tape has a volume serial of TRCTAP and resides on a 3400 tape drive. Note that the GTFPARM parmlib member is used to specify the trace options.

Example: Start GTF, Trace Output to Existing Data Set on Tape

```
START GTF,3400,TRCTAP,(MODE=EXT),DISP=OLD,DSNAME=TPOUTPUT

AHL103I TRACE OPTIONS SELECTED--SYSM,DSP,PCI,SRM,TRC,USR

00 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

REPLY 00,U

AHL031I GTF INITIALIZATION COMPLETE
```

Starting GTF with Trace Options Stored in SYS1.PARMLIB

This example shows how to store trace options in a member of SYS1.PARMLIB. This can save you time when starting GTF. First store one or more combinations of trace options as members in SYS1.PARMLIB, and include a SYSLIB DD statement in the cataloged procedure. When you start GTF, GTF will then retrieve the trace options from SYS1.PARMLIB, instead of prompting the operator to supply them through the console. GTF displays the trace options for you, and then issue message AHL125A, to which the operator replies **U** to accept the parmlib options.

Example: Starting GTF with Trace Options Stored in SYS1.PARMLIB

This example shows the job control statements and utility JCL statements needed to add trace options to SYS1.PARMLIB using IEBUPDTE:

```
//GTFPARM    JOB     MSGLEVEL=(1,)
//           EXEC    PGM=IEBUPDTE,PARM=NEW
//SYSPRINT   DD      SYSOUT=A
//SYSUT2     DD      DSNAME=SYS1.PARMLIB,DISP=SHR
//SYSIN      DD      DATA
./       ADD    NAME=GTFA,LIST=ALL,SOURCE=0
TRACE=SYSP,USR
SVC=(1,2,3,4,10),IO=(D34,D0C),SSCH=ED8,PI=15
./       ADD    NAME=GTFB,LIST=ALL,SOURCE=0
TRACE=IO,SSCH,TRC
./       ADD    NAME=GTFC,LIST=ALL,SOURCE=0
TRACE=SYS,PCI
/*
```

References

- See *z/OS DFSMSdfp Utilities* for descriptions of the statements.
- See *z/OS MVS JCL Reference* for descriptions of the statements.
- See *z/OS MVS Initialization and Tuning Reference* for further information about SYS1.PARMLIB.

A sample SYSLIB DD statement to be included in a GTF cataloged procedure might look like this:

```
//SYSLIB DD DSN=SYS1.PARMLIB(GTFA),DISP=SHR
```

The new member name can also be specified on the START command while using the IBM-supplied GTF procedure, as in the following example:

```
S GTF,,,(MODE=EXT,TIME=YES),MEMBER=GTFB
```

Starting GTF Without Trace Options in a Member

The following example show an installation-written procedure where there is no predefined member with trace options specified. The procedure contains no SYSLIB DD statement. When GTF is started with a procedure containing no SYSLIB DD statement, the operator receives message AHL100A to prompt for GTF trace options.

In this example, an installation-written cataloged procedure, USRPROC, is invoked to start GTF in external mode to a direct access data set, ABCTRC, on device 250. The trace options selected by the operator result in trace data being gathered for:

- All SVC and IO interruptions
- All SSCH operations
- All matching SLIP traps with a tracing action specified or SLIP traps in DEBUG mode
- All dispatcher events
- All issuers of the GTRACE macro will have their user data recorded in the trace buffers.

The trace data is written into the data set ABCTRC. (Note that when the end of the primary extent is reached, writing continues at the beginning.)

Example: Starting GTF Without Trace Options in a Member

```
START USRPROC,250,333005,(MODE=EXT),DSN=ABCTRC


00 AHL100A SPECIFY TRACE OPTIONS

REPLY 00,TRACE=SVC,SSCH,IO,DSP,SLIP,USR

AHL103I TRACE OPTIONS SELECTED--USR,DSP,SVC,IO,SLIP,SSCH

01 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

REPLY 01,U

AHL031I GTF INITIALIZATION COMPLETE
```

Starting GTF to Trace VTAM Remote Network Activity

GTF can trace VTAM activity only if VTAM is started with the GTF option. See *z/OS Communications Server: SNA Operation* for details. In the following example, GTF options are not stored in parmlib; the operator enters the trace options directly at the console. Three GTF options are required to record all VTAM traces:

- RNIO must be specified so that the VTAM I/O trace can function for an NCP or a remote device attached to the NCP.
- IO or IOP must be specified so that the VTAM I/O trace can function for a local device.
- USR or USRP must be specified so that the VTAM buffer and the NCP line traces can function.

The operator must enter the START GTF command before a trace can be activated from VTAM.

Example: Starting GTF to Trace VTAM Remote Network Activity

```
START MYPROC.EXAMPLE8,,,(MODE=EXT)

00 AHL100A SPECIFY TRACE OPTIONS

REPLY 00,TRACE=RNIO,IO,USRP

AHL103I TRACE OPTIONS SELECTED--IO,USRP,RNIO

01 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

REPLY 01,USR=(FF0,FF1),END

AHL031I GTF INITIALIZATION COMPLETE
```

Stopping GTF

The operator can enter the STOP command at any time during GTF processing. The amount of time you let GTF runs depends on your installation and the problem you are trying to capture, but a common time is between 15 and 30 minutes.

To stop GTF processing, have the operator enter the STOP command. The STOP command must include either the GTF identifier specified in the START command, or the device number of the GTF trace data set if you specified MODE=EXT or MODE=DEFER to direct output to a data set.

If you are not sure of the identifier, or the device number of the trace data set, ask the operator to enter the following command:

```
DISPLAY A,LIST
```

The operator must enter the STOP command at a console with master authority.

Example: DISPLAY A,LIST output

In the following example of DISPLAY A,LIST output, the identifier for GTF is EVENT1.

```
IEE114I 14.51.49 1996.181 ACTIVITY     FRAME LAST   F      E   SYS=SY1
 JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
00000    00003    00000      00016    00000    00000/00000       00000
 LLA      LLA      LLA      NSW  S  VLF      VLF      VLF      NSW  S
 JES2     JES2     IEFPROC  NSW  S
 GTF      EVENT1   IEFPROC  NSW  S
```

The general format of the STOP command is as follows:

{STOP|P} ***identifier***

Example: Stopping GTF

To stop GTF for the identifier EVENT1, enter the command:

```
STOP EVENT1
```

When the STOP command takes effect, the system issues message AHL006I. If the system does not issue message AHL006I, then GTF tracing continues, remaining active until a STOP command takes effect or the next initial program load (IPL). When this happens, you will not be able to restart GTF tracing. In this case, you can use the FORCE ARM command to stop GTF.

If there were several functions started with the same identifier on the START command, using the same identifier on the STOP command will stop all those functions.

Reference

See *z/OS MVS System Commands* for more information about the STOP and FORCE ARM commands.

Example of Stopping GTF

Example: Stopping GTF Using an Identifier

This example starts GTF tracing with the identifier EXAMPLE and with trace data maintained in the GTF address space. The DSN keyword is entered to prevent allocation of an external trace data set as specified in the cataloged procedure.

```
START GTF.EXAMPLE,,,(MODE=INT),DSN=NULLFILE
```

The command to stop the GTF tracing started above is:

```
STOP EXAMPLE
```

Example: Displaying Active Jobs to Stop GTF

This example shows an instance when you would need to display active jobs before stopping GTF. The example starts GTF tracing with trace data recorded on an external device, data set GTF.TEST01. Note that you do not have to specify MODE=EXT, because it is the default.

```
S GTF,,,DSNAME=GTF.TEST01,VOLUME=SER=IPCS01,DISP=OLD
```

Because it is not apparent which is the GTF recording device, the operator has to display active jobs with the DISPLAY A,LIST command before stopping GTF. In this example, the device number for GTF is 0227:

```
IEE114I 09.33.45 1996.183 ACTIVITY 951
 JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
00001    00006    00001      00015    00002    00001/00300       00000
 LLA      LLA      LLA      NSW  S  VLF      VLF      VLF      NSW  S
 VTAM     VTAM     VTAM     NSW  S  JES2     JES2     IEFPROC  NSW  S
 TCAS     TCAS     TSO      OWT  S  SDJSST1B STEP1             OWT  J
 GTF      0227     IEFPROC  NSW  S
```

The operator could then stop GTF using the device number 227 as follows:

```
STOP 227
```

GTF Trace Options

This topic describes the GTF options you can specify through either system prompting in response to the START GTF command or in a predefined parmlib or data set member. However, GTF will not use certain combinations of options; see Table 10-1 on page 10-23 for a list of those combinations.

Some GTF trace options also require keywords. If you specify options requiring keywords in the member or data set containing the predefined options, it must also contain the associated keywords. These are explained in “Prompting Keywords” on page 10-23.

ASIDP

Requests that GTF tracing be limited to a subset of address spaces. ASIDP requests GTF prompting for one to five address space identifiers (ASID) in which you want GTF tracing to occur. ASIDP works only with a GTF option that generates tracing, such as SVC or IO. For information about responding to GTF prompts, see “Prompting Keywords” on page 10-23.

CCW
Requests tracing of channel programs and associated data for I/O events. CCW is valid only if the other trace options include SSCH, SSCHP, IO, or IOP. See Table 10-1 on page 10-23.

CCWP
Requests tracing of channel programs and associated data for I/O events, and requests GTF prompting for the following information:

- Tracing channel command words (CCW) for start subchannel (SSCH) operations or I/O interruptions or both
- Maximum number of CCWs for each event
- Maximum number of bytes of data for each CCW
- Optional input/output supervisor block (IOSB) and error recovery procedure work area (EWA) tracing
- Size of the program controlled interrupt (PCI) table

For information about responding to GTF prompts, see “Prompting Keywords” on page 10-23.

CCWP is valid only if the other trace options include SSCH, SSCHP, IO, or IOP. See Table 10-1 on page 10-23.

CSCH
Requests recording for all clear subchannel operations. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

DSP
Requests recording for all dispatchable units of work: service request block (SRB), local supervisor routine (LSR), task control block (TCB) and Supervisor Call (SVC) prolog dispatch events. If the operator specifies both the SYSM and DSP trace options, GTF records minimal trace data for DSP. Otherwise, GTF records comprehensive trace data for DSP.

EXT
Requests comprehensive recording for all external interruptions. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

HSCH
Requests recording for all halt subchannel operations. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

IO
Requests recording of all non-program-controlled I/O interruptions. Unless you also have the operator specify the PCI trace option, GTF does not record program-controlled interruptions. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

IOX
Requests recording of all non-program-controlled I/O interruptions providing a summary of a complete channel program for the I/O interruption in an I/O summary trace record. Unless you also have the operator specify the PCI trace option, GTF does not record program-controlled interruptions.

IOP

Requests GTF prompting for specific device numbers for which you want GTF to record non-program-controlled I/O interruptions. Unless you also have the operator specify the PCI trace option, GTF does not record program-controlled interruptions. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options. For information about responding to GTF prompts, see “Prompting Keywords” on page 10-23.

IOXP

Requests GTF prompting for specific device numbers for which you want GTF to record non-program-controlled I/O interruptions providing a summary of a complete channel program for the I/O interruption in an I/O summary trace record. Unless you also have the operator specify the PCI trace option, GTF does not record program-controlled interruptions. For more information on responding to GTF prompts, see “Prompting Keywords” on page 10-23.

If an installation chooses to specify either IO or IOP in addition to IOX or IOXP, they will receive IOX records for DASD and tape devices and IO records for all other devices.

JOBNAMEP

Requests that GTF tracing be limited to a subset of jobs. JOBNAMEP requests GTF prompting for one through five job names for which you want GTF tracing to occur.

These job names can be generic, as well as specific, job names. If you want to specify generic job names, the operator must use * or % in the job name.

The asterisk is a placeholder for one or more valid job name characters, or indicates no characters. For example, if the operator enters JOBNAMEP=I*MS*, GTF will process trace data for address spaces with job names IABMS01, IAMS, IMS, IMSA, IMS00012, and so on. However, if the operator enters JOBNAMEP=*MASTER*, that job name represents only the master address space.

The percent symbol is a placeholder for a single valid job name character. For example, if the operator enters JOBNAMEP=I%MS%%, GTF will process trace data for address spaces with job names IAMS01 and IXMSBC, but not job names IMS001 or I999MS. The combination %* is a placeholder for at least one character.

JOBNAMEP works only with a GTF option that generates tracing, such as SVC or IO. For information on responding to GTF prompts, see “Prompting Keywords” on page 10-23.

MSCH

Requests recording for all modify subchannel operations. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

PCI

Requests recording of intermediate status interruptions in the same format as other I/O trace records that GTF creates. Specifically, PCI causes GTF to record program-controlled I/O interruptions, initial status request interruptions, resume subchannel operation instruction, and suspend channel program interruptions. When the operator selects specific devices as a result of prompting for I/O events (IOP trace option), GTF records intermediate status interruptions for only those devices. PCI is valid only when the other trace options include IO, IOP, SYS, SYSM, or SYSP.

PI
Requests comprehensive recording for all program interruptions (0-255). See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

PIP
Requests GTF prompting for those interruption codes for which you want GTF to record program interruptions. For information about responding to GTF prompts, see “Prompting Keywords” on page 10-23. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

RNIO
Requests recording of all Virtual Telecommunications Access Method (VTAM) network activity. If the operator specifies both the SYSM and RNIO trace options, GTF will record minimal trace data for RNIO. Otherwise, GTF records comprehensive trace data for RNIO.

RR
Requests comprehensive recording of data associated with all invocations of recovery routines (such as STAE and ESTAE routines). GTF creates a trace record describing the activity of the recovery routine when control passes from the recovery routine back to the recovery termination manager (RTM). See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

{SIO|SIOP}
If the operator specifies the SIO or SIOP trace option, GTF processes that request as a request for SSCH or SSCHP. GTF issues message AHL138I to indicate this substitution. Subsequent messages refer to the original SIO or SIOP trace option.

Note: The SIO keyword is provided only for compatibility; it is recommended that you use the SSCH keyword instead. The SIOP option is provided only for compatibility; it is recommended that you use the SSCHP option instead.

SLIP
Requests that a trace entry be made each time:
- A match occurs for a SLIP trap with ACTION=TRACE
- A SLIP trap with the SLIP DEBUG option is checked

The amount of data and the type of SLIP trace record to be built is specified on the SLIP command.

SRM
Requests recording of trace data each time the system resource manager (SRM) is invoked. If the operator specifies both the SYSM and SRM trace options, GTF records minimal trace data for SRM. Otherwise, GTF records comprehensive trace data for SRM.

SSCH
Requests recording for start subchannel and resume subchannel operations. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

SSCHP
Requests GTF prompting for the specific device numbers for which you want GTF to record start subchannel and resume subchannel operations. For

information about responding to GTF prompts, see “Prompting Keywords” on page 10-23. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

SVC
Requests comprehensive recording for all SVC interruptions. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

SVCP
Requests GTF prompting for those SVC numbers for which you want data recorded. For information about responding to GTF prompts, see “Prompting Keywords” on page 10-23. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

SYS
Requests recording of comprehensive trace data for all of the following:
- Clear subchannel operations
- External interruptions
- Halt subchannel operations
- I/O interruptions
- Modify subchannel operations
- Program interruptions
- Recovery routines
- Start subchannel and resume channel operations
- SVC interruptions.

Because specifying SYS automatically causes GTF to trace all of these events, GTF will ignore the following trace options if the operator specifies them in any form: CSCH, HSCH, MSCH, SSCH, EXT, IO, PI, RR, SVC. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

SYSM
Requests recording of minimal trace data for the same events as SYS.

Because specifying SYSM automatically causes GTF to trace all of these events, GTF will ignore the following trace options if the operator specifies them in any form: CSCH, HSCH, MSCH, SSCH, EXT, IO, PI, RR, SVC.

If the operator specifies DSP, RNIO, or SRM in addition to SYSM, GTF produces minimal, rather than comprehensive, trace data for those events.

SYSP
Requests recording for the same events as the SYS option, but causes GTF to prompt for selection of specific SVC, IO, SSCH, and PI events that you want recorded. For information about responding to GTF prompts, see “Prompting Keywords” on page 10-23.

Because specifying SYSP automatically causes GTF to trace all of these events, GTF will ignore the following trace options if the operator specifies them in any form: CSCH, HSCH, MSCH, SSCH, EXT, IO, PI, RR, SVC. See Table 10-1 on page 10-23 for more information on combining this option with other GTF options.

TRC
Requests recording of those trace events that are associated with GTF itself. Unless you request TRC, GTF will not trace these events. TRC works only with a GTF option that generates tracing, such as SVC or IO.

USR

Requests recording of all data that the GTRACE macro passes to GTF. The operator must specify USR or USRP to trace data from the GTRACE macro. Use USRP for specific events. If USR is used instead of USRP, the trace data set might be full of unwanted records. When you code the GTRACE macro but do not specify USR or USRP, GTF ignores the GTRACE macro. See Table 10-1 for more information on combining this option with other GTF options.

Reference

See *z/OS MVS Programming: Assembler Services Reference ABE-HSP* for information about coding the GTRACE macro.

USRP

Requests GTF prompting for specific event identifiers (EID) of the data that the GTRACE macro passes to GTF. The EIDs represent user, program product, or IBM subsystem and component events. See Table 10-4 on page 10-29 for a list of EID values.

See Table 10-1 for more information on combining this option with other GTF options. For information about responding to GTF prompts, see "Prompting Keywords".

Combining GTF Options

Table 10-1 shows those TRACE options that GTF will *not* use in combination. If the operator specifies two or more options from the same row, GTF uses the option that has the lower column number and ignores the other options. For example, if the operator specifies both SYSP and PI (see row D), GTF uses SYSP (column 2) and ignores PI (column 5).

Table 10-1. Combining GTF Options

Row	Columns				
	1	2	3	4	5
A	SYSM	SYSP	SYS	SSCHP	SSCH
B	SYSM	SYSP	SYS	IOP, IOXP	IO, IOX
C	SYSM	SYSP	SYS	SVCP	SVC
D	SYSM	SYSP	SYS	PIP	PI
E	SYSM	SYSP	SYS	EXT	
F	SYSM	SYSP	SYS	RR	
G	SYSM	SYSP	SYS	CSCH	
H	SYSM	SYSP	SYS	HSCH	
I	SYSM	SYSP	SYS	MSCH	
J	CCWP	CCW			
K	USRP	USR			

If an installation chooses to specify either IO or IOP in addition to IOX or IOXP, they will receive IOX records for DASD and tape devices and IO records for all other devices.

Prompting Keywords

When the operator specifies any of the trace options listed in Table 10-2 on page 10-24, GTF prompts for specific values by issuing message AHL101A:

```
AHL101A SPECIFY TRACE EVENT KEYWORDS - keyword=,...,keyword=
```

The keywords issued in the message correspond to the trace options specified. Enter only the trace event keywords appearing in the message text. The trace options and their corresponding keywords are:

Table 10-2. GTF Trace Options and Corresponding Prompting Keywords

Trace Option	Prompting Keyword	Number of Prompting Values Allowed
ASIDP	ASID=	5
CCWP	CCW=	N/A
IOP, IOXP, SYSP	IO=SSCH=	128
IOP, IOXP, SSCHP, SYSP	IO=SSCH=	128
JOBNAMEP	JOBNAME=	5
PIP, SYSP	PI=	50
SSCHP, SIOP, SYSP	SIO=	128
SSCHP, SIOP, SYSP	SSCH=	128
SVCP, SYSP	SVC=	50
USRP	USR=	50
Note: The SIO keyword is provided only for compatibility; it is recommended that you use the SSCH keyword instead. The SIOP option is provided only for compatibility; it is recommended that you use the SSCHP option instead.		

Guidelines for Specifying Values for Prompting Keywords: Use the following guidelines when replying to message AHL101A for prompting keywords:

- If you do not specify a reply for each of the keywords displayed in message AHL101A, GTF records all the events for that trace option, which increases the amount of storage that GTF requires. IBM recommends that you specify values for each keyword displayed, selecting the values that will help you debug your problem.
- You can only enter values for keywords displayed in message AHL101A.
- GTF limits the number of specific values that the operator can supply through prompting, see Table 10-2 for the maximum number of values allowed for each keyword. If you specify more than the maximum values, GTF issues a message to which the operator replies by respecifying values for all appropriate keywords.
- Keep in mind that prompting increases the amount of storage that GTF requires, because storage requirements depend on the trace options you specify. See “Determining GTF’s Storage Requirements” on page 10-9 for further information.
- Within a given reply, each keyword that the operator specifies must be complete. If you need more values for a keyword than will fit into one reply, the operator repeats the keyword in the next reply, and codes the additional values for that keyword. The following are examples of correct replies:

  ```
  REPLY 01 IO=(191-193),SVC=(1,2,3,4,5)
  REPLY 01 SVC=(6,7,8,9,10)
  ```

  The maximum number of values that GTF allows for a keyword does not change, regardless of whether the operator enters one or more replies to specify all the values for the keyword.
- After supplying all keywords and values, the operator must enter the END keyword, which signifies that the event definition is complete. If the system does not find the END keyword in a reply, the system issues message AHL102A to

prompt for additional event keywords and values. When the system finds the END keyword, the system issues message AHL103I to list all of the trace options that are in effect.

For sample prompting sequences, see “Examples of Sample Prompting Sequences” on page 10-29.

Use the following keywords when GTF prompts for values by issuing message AHL101A:

ASID=(*asid1*[,*asidn*]...[,*asid5*])
Specifies one through five identifiers for address spaces in which you want GTF tracing to occur. The values ‘asid1’ through ‘asid5’ are hexadecimal numbers from X'0001' to the maximum number of entries in the address space vector table (ASVT). If the operator specifies ASIDP, but does not specify ASID= before replying END, GTF traces all address space identifiers.

If the number of values for ASIDP requires more than one line, and a particular ASID value is incorrect, GTF allows the operator to respecify the correct value without having to reenter all ASIDs.

If the operator specifies both ASIDP and JOBNAMEP, GTF will trace address spaces that ASIDP did not identify, if some of the jobs that JOBNAMEP identifies run in other address spaces.

CCW=([S|I|SI][,CCWN=*nnnnn*][,DATA=*nnnnn*][,IOSB][,PCITAB=n])
Specifies different options for tracing channel programs. If the operator specifies CCW= more than once, GTF uses the last specification of CCW=.

If the operator specifies CCWP, but does not specify a value for keyword CCW=, GTF’s default CCW tracing depends on what other trace options were specified. The following table shows the defaults for CCW tracing depending on other trace options specified:

Table 10-3. CCW Defaults for Selected TRACE Options

Other Trace Options Selected	CCW Subparameter Defaults
SSCH or SSCHP	S
IO or IOP or IOX or IOXP	I
SSCH or SSCHP or IO or IOP or IOX or IOXP	SI
PCI	PCITAB=1
SSCH or SSCHP or IO or IOP or IOX or IOXP	CCWN=50
SSCH or SSCHP or IO or IOP or IOX or IOXP	DATA=20

```
Examples:
TRACE=IO,CCWP            CCW defaults to: CCW=(I,CCWN=50,DATA=20)
TRACE=IOP,SSCH,PCI,CCWP  CCW defaults to: CCW=(SI,CCWN=50,DATA=20,PCITAB=1)
```

If the operator specifies an option more than once in one line, GTF uses the last specification of that option. An exception is that GTF uses the first specification of S, I, or SI. If a line contains an error, GTF prompts the operator to respecify the value.

S|I|SI
Specifies the type of I/O event for which you want channel programs traced. If the operator specifies more than one option, GTF uses the first option.

S Specifies GTF tracing of channel programs for start subchannel and resume subchannel operations. CCW=S works only with the SSCH or SSCHP trace options.

I Specifies GTF tracing of channel programs for I/O interruptions, including program-controlled interruptions if the operator specifies PCI as a trace option. CCW=I works only with the IO or IOP trace options.

SI Specifies GTF tracing of channel programs for start subchannel and resume subchannel operations and I/O interruptions. CCW=SI works only with either SSCH or SSCHP and either IO or IOP as trace options.

CCWN=*nnnnn*
Specifies the maximum number of CCWs traced for each event. The value *nnnnn* is any decimal number from 1 to 32767. The default is 50.

DATA=*nnnnn*
Specifies the maximum number of bytes of data traced for each CCW. The value *nnnnn* is any decimal number from 0 to 32767. The default is 20.

GTF traces *nnnnn* bytes of data for each CCW on the data chain. GTF also traces *nnnnn* bytes of data for each word in an indirect data addressing word (IDAW) list.

For start subchannel or resume subchannel operations, GTF does not trace data for read, read backwards, or sense commands in the channel programs. If no data is being transferred, regardless of the type of I/O operation, GTF does not trace data for read, read backwards, or sense commands.

When the data count in the CCW is equal to or less than *nnnnn*, GTF traces all data in the data buffer. When the data count in the CCW is greater than *nnnnn*, GTF traces data only from the beginning and end of the data buffer. If you examine the traced data, you can tell whether the channel completely filled the buffer on a read operation.

GTF uses a different CCW tracing method for a data transfer that is in progress when an I/O interruption occurs. Instead of using the data count in the CCW, GTF tracing depends on the transmitted data count. The transmitted data count is the difference between the data count in the CCW and the residual count in the channel status word (CSW).

- When the residual count in the CSW is greater than the data count in the CCW, then GTF traces all of the data in the CCW.
- When the transmitted data count is less than or equal to *nnnnn*, GTF traces all of the transmitted data.
- When the transmitted data count is greater than *nnnnn*, GTF traces data only from the beginning and end of the transmitted data.

IOSB
Specifies tracing of the input/output supervisor block (IOSB) and, if available, the error recovery procedure work area (EWA), for all CCW events. If you do not specify IOSB, then GTF performs IOSB and EWA tracing only if GTF encounters an exceptional condition when tracing a channel program.

PCITAB=*n*
Specifies a decimal number of 100-entry increments for GTF to allocate in an internal program-controlled interruption (PCI) table. The value of *n* is an integer from 1 to 9. The default is 1 (100 entries).

The PCI table keeps track of the channel programs that use PCI. One entry in the PCI table contains information about a program-controlled interruption in one channel program. An entry in the PCI table includes a CCW address and an IOSB address.

IO=(*DEVCLASS=xxxx,DEVCLASS=xxxx,devnum1* [*,devnumn...,devnum128*])
Specifies devices for which you want I/O interruptions traced.

Devices are specified by entering a device number or a device class.

The device number must be specified in hexadecimal and is not the same as the subchannel number. If the operator specifies any combination of IO= and SSCH=, and IO=SSCH=, the combined number of device numbers for all prompting keywords cannot exceed 128. Specify device numbers individually, or as a range of device numbers with a dash (-) or colon (:) separating the lowest and highest number in the range. For example, to trace I/O interruptions for device numbers 193 through 198, you specify IO=(193-198).

The device class must be specified with the DEVCLASS= keyword parameter, which provides the ability to trace all devices in the specified device class. The allowable keyword parameters are:

```
TAPE  (magnetic tape devices)
COMM  (communications)
DASD  (direct access storage device)
DISP  (display)
UREC  (unit/record)
CTC   (channel to channel)
```

If the operator specifies IOP, IOXP, or SYSP and does not specify IO= in the response to the prompting messages, GTF processing proceeds as if the operator specified IO, IOX or SYS event keywords respectively.

For the following examples, the I/O device numbers and associated device types listed below are used:

I/O Device Number	Device Type
230	3390 DASD
450	3490 Tape Drive
575	3480 Tape Drive
663	3380 DASD
020	3274 Communications Controller

Example 1:

```
IO=(DEVCLASS=DASD,450)
```

the resulting trace includes information for all DASD devices and one 3490 tape drive at address 450.

Example 2:

```
IO=(DEVCLASS=DASD,DEVCLASS=TAPE,020)
```

the resulting trace includes information for all DASD and TAPE devices and the communications controller at address 020.

Example 3:

```
IO=(450-663)
```

the resulting trace includes information for the devices at addresses 450, 575, and 663.

IO=SSCH=(*DEVCLASS=xxxx,DEVCLASS=xxxx,*devnum1[,devnumm.... [,devnum128])
Specifies devices for which you want both I/O interruptions and start subchannel operations traced.

See the IO= prompting keyword for a description of how to specify devices to be traced.

JOBNAME=(*jobname1*[,*jobnamen*]...[,*jobname5*])
Specifies one through five job names for which you want GTF tracing to occur. The values *job1* through *job5* must be valid job names.

These job names can be generic, as well as specific, job names. If you want to specify generic job names, the operator must use * or % in the job name.

The asterisk is a placeholder for one or more valid job name characters, or indicates no characters. For example, if the operator enters JOBNAMEP=I*MS*, GTF will process trace data for address spaces with job names IABMS01, IAMS, IMS, IMSA, IMS00012, and so on. However, if the operator codes JOBNAMEP=*MASTER*, that job name represents only the master address space.

The percent symbol is a placeholder for a single valid job name character. For example, if the operator codes JOBNAMEP=I%MS%%, GTF will process trace data for address spaces with job names IAMS01 and IXMSBC, but not job names IMS001 or I999MS. The combination %* is a placeholder for at least one character.

If the operator specifies JOBNAMEP, but does not specify JOBNAME before replying END, GTF traces all job names.

If the number of values for JOBNAMEP requires more than one line, and a particular job name value is incorrect, GTF allows the operator to respecify the correct value without having to reenter all job names.

If the operator specifies both ASIDP and JOBNAMEP, GTF will trace jobs that JOBNAMEP did not identify, if some of the address spaces that ASIDP identifies contain jobs that JOBNAMEP did not identify.

PI=(*code0*[,*coden*]...[,*code50*])
Specifies 1 through 50 program interruption codes, in decimal notation, that you want traced. If the operator specifies PIP or SYSP, and does not specify PI= in response to this prompting message, GTF traces all program interruptions.

SSCH=(*DEVCLASS=xxxx,DEVCLASS=xxxx,devnum1*[,*devnumn....,* devnum128])
Specifies devices for which you want start subchannel operations traced.

See the IO= prompting keyword for a description of how to specify devices to be traced.

SVC=(*svcnum1*[,*svcnumn*]...[,*svcnum50*])
Specifies 1 through 50 SVC numbers, in decimal notation, that you want traced. If the operator specifies SVCP or SYSP, and does not specify SVC= in response to the prompting message, GTF traces all SVC numbers.

USR=(*event1*[,*eventn*]...[,*event50*])
Specifies 1 through 50 user event identifiers (EIDs) for which you want user data traced. The values for USR are three-digit hexadecimal numbers, as follows:

Table 10-4. Event Identifiers and the Types of Events They Represent

Identifier (Hex)	Type of Event
000-3FF	User
400-5FF	Reserved for program products
600-FFF	Reserved for IBM subsystems and components

If the operator specifies USRP and does not specify USR= in response to the prompting message, all instances of GTRACE issued with TEST=YES will return with an indication that tracing is not active.

Examples of Sample Prompting Sequences

This example shows how to store prompting keywords in a SYS1.PARMLIB member.

If you start GTF with options requiring prompting keywords stored in SYS1.PARMLIB, these prompting keywords must also appear in the parmlib member. If prompting keywords are used in the parmlib member without the replies included, GTF will not obtain the replies from the console. GTF will use the options without the prompting (for example, SVCP becomes SVC). A SYSLIB DD statement in your cataloged procedure causes GTF to read the prompting keywords from the specified parmlib member. The second and subsequent logical records in the member should contain only the prompting keywords.

GTF uses either the END keyword, or end-of-file on the member as the indicator that there is no more prompting input from parmlib. If the number of events for one keyword require more than one record, respecify the keyword in a subsequent prompting record with the additional events, as follows:

```
Record #1 TRACE=IOP,SVCP,SSCH

Record #2 IO=(D34,D0C),SVC=(1,2,3)

Record #3 SVC=(4,5,6,7,8,9,10),END
```

Do not respecify the keyword through the system console at this point, because GTF will then override all of the options and keywords in the parmlib member.

When GTF finishes reading the options and prompting keywords in the parmlib member, it displays the options through message AHL103I:

```
AHL103I TRACE OPTIONS SELECTED--IOP,SVCP,SSCH
AHL103I IO=(D34,D0C),SVC=(1,2,3,4,5,6,7,8,9,10)
```

This message may be a multiple-line message, depending on the number of options the operator selects. If the set of devices specified for IO= and SSCH= are identical, message AHL103I will show them as if specified by use of IO=SSCH.

After GTF displays all of the options specified, the operator then has the opportunity to accept the parmlib options, or completely change the options by respecifying them through the console by replying to the following message:

```
AHL125A RESPECIFY TRACE OPTIONS OR REPLY U.
```

Example: Specifying Prompting Trace Options SYSP and USRP

In this example, the operator started GTF in external mode to the data set defined in the cataloged procedure.

The operator selected two trace options in reply 00:
- SYSP requests that GTF trace specific system event types.
- USRP requests that GTF trace specific user entries that the GTRACE macro generates.

Message AHL101A instructed the operator to specify values for the SVC, IO, SSCH, PI, and USR keywords.

In reply 01 to message AHL101A, the operator selected:
- Five SVCs
- Two devices for non-program-controlled I/O interruptions
- One device for SSCH operations
- Three user event identifiers.

GTF does not record any other SVC, IO, and SSCH events. Because the operator did not specify any program interruption codes for PI=, GTF would trace all program interruptions.

Specifying Prompting Trace Options SYSP and USRP

```
START MYPROC.EXAMPLE7,,,(MODE=EXT)

00 AHL100A SPECIFY TRACE OPTIONS

REPLY 00,TRACE=SYSP,USRP

01 AHL101A SPECIFY TRACE EVENT KEYWORDS--IO=,SSCH=,SVC=,PI=,USR=

01 AHL101A SPECIFY TRACE EVENT KEYWORDS--IO=SSCH=

REPLY 01,SVC=(1,2,3,4,10),IO=(191,192),USR=(10,07A,AB)

02 AHL102A CONTINUE TRACE DEFINITION OR REPLY END

REPLY 02,SSCH=282,END

AHL103I TRACE OPTIONS SELECTED--SYSP,PI,IO=(191,192),SSCH=(282)

AHL103I SVC=(1,2,3,4,10),USR=(010,07A,0AB)

03 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

REPLY 03,U
```

Example: Specifying Prompting Trace Options

In this example, the operator started GTF in external mode, using the trace options defined in the data set specified in the cataloged procedure. The operator is then prompted for information as follows:
- Message AHL100A prompts the operator for trace options.
- In reply 00, the operator selected six trace options: SSCHP, IOP, PCI, CCWP, SVC, and JOBNAMEP.
- Message AHL101A prompts the operator to specify values for the IO, SSCH, CCW and JOBNAME prompting keywords.

- In reply 01, the operator selects one device for tracing both IO and SSCH events and limits GTF tracing to one job.
- In reply 02, the operator specifies five options for CCW tracing.

The final result of the operator's specifications is that GTF traces CCWs for both start subchannel operations and I/O interruptions at device 580 for the job BACKWARD, and all SVCs in BACKWARD's address space. GTF would allocate 200 entries in the PCI table, and trace up to 100 CCWs, up to 40 bytes of data for each CCW, and the IOSB.

Specifying Prompting Trace Options

```
START USRPROC,,,(MOD=EXT)

00 AHL100A SPECIFY TRACE OPTIONS

REPLY 00, TRACE=SSCHP,IOP,PCI,CCWP,SVC,JOBNAMEP

01 AHL101A SPECIFY TRACE EVENT KEYWORDS
        --IO=,SSCH=,CCW=,JOBNAME=,IO=SSCH=

REPLY 01,JOBNAME=(BACKWARD),IO=SSCH=580

02 AHL102A CONTINUE TRACE DEFINITION OR REPLY END

REPLY 02,CCW=(CCWN=100,DATA=40,PCITAB=2,IOSB,SI),END

AHL103I TRACE OPTIONS SELECTED--PCI,SVC,IO=SSCH=(580)

AHL103I CCW=(SI,IOSB,CCWN=100,DATA=40,PCITAB=2)

AHL103I JOBNAME=(BACKWARD)

03 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

REPLY 03,U
```

Receiving GTF Traces

GTF writes trace data in GTF trace tables in the GTF address space in storage. GTF trace data in storage are printed or viewed as part of a dump, if the dump options list includes TRT to request trace data. The following table shows the dumps that have TRT in their default options. For unformatted dumps that are printed or viewed through IPCS, format the trace data by specifying the IPCS GTFTRACE subcommand or using the IPCS Trace Processing selection panel.

To format and print GTF trace data in a GTFOUTxx or IEFRDER data set, specify the IPCS GTFTRACE subcommand or use the IPCS Trace Processing selection panel.

If the GTF data was created for VTAM diagnosis, you can use the ACF/TAP program to format the VTAM data.

Dump	How to Obtain Trace Data
ABEND dump to SYSABEND	Default
ABEND dump to SYSMDUMP	Not available
ABEND dump to SYSUDUMP	Request SDATA=TRT

Dump	How to Obtain Trace Data
SNAP dump	Request SDATA=TRT
Stand-alone dump	Default
SVC dump for SDUMP or SDUMPX macro	Default
SVC dump for DUMP operator command	Default
SVC dump for SLIP operator command with ACTION=SVCD, ACTION=STDUMP, ACTION=SYNCSVCD, or ACTION=TRDUMP	Default
Any dump customized to exclude trace data	Request SDATA=TRT

References

- See *z/OS MVS IPCS Commands* for the GTFTRACE subcommand.
- See *z/OS MVS IPCS User’s Guide* for the panel interface.

Combining, Extracting, and Merging GTF Trace Output

GTF trace data can be combined with other data or extracted from dumps and data sets using two IPCS subcommands: COPYTRC and MERGE.

Use consolidated or merged trace output to show the chronology of events around the time of an error. Specify start and stop times for the merge to see events beginning a little before the error occurred and ending a little after. On the CTRACE and GTFTRACE subcommands, specify the jobs and address space identifiers (ASID) involved, so that the merged output contains only pertinent trace records.

Merging is most useful when several components are running traces; the system can also be running a GTF trace. Each component puts its trace records into its own buffers independently. GTF is independent from all of the component traces. You can merge these separate records into one chronological sequence to make diagnosis easier.

Reference

See *z/OS MVS IPCS Commands* for more information about COPYTRC and MERGE.

Combining and Extracting GTF Output

Use the IPCS COPYTRC subcommand to do one or more of the following:

- Consolidate GTF trace data into one data set from:
  - Multiple GTF data sets
  - Multiple GTF data sets, dumps, or both
  - More than one system
- Extract GTF trace data from SVC dumps and stand-alone dumps
- Extract from merged data the GTF trace data for a specified list of systems

Example: Consolidating GTF Output from Multiple Data Sets

If you have GTF set up to write data for one system to multiple data sets, you can use the IPCS COPYTRC subcommand to consolidate the data into one data set. You should do this before you consolidate GTF data from multiple systems with the MERGE or COPYTRC subcommands.

A GTF cataloged procedure with 3 data sets defined for GTF data from system SYS01 might look like the following:

```
//GTFABC    PROC   MEMBER=GTFPARM
//IEFPROC   EXEC   PGM=AHLGTF,REGION=2880K,TIME=1440,
//                 PARM=('MODE=EXT,DEBUG=NO')
//IEFRDER   DD     DSNAME=SYS1.GTFTRC,UNIT=SYSDA,
//                 SPACE=(4096,20),DISP=(NEW,KEEP)
//SYSLIB    DD     DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
//GTFOUT1   DD     DSNAME=SYS01.DSN1,UNIT=&DEVICE,DISP=&DSPS;
//GTFOUT2   DD     DSNAME=SYS01.DSN2,UNIT=&DEVICE,DISP=&DSPS;
//GTFOUT3   DD     DSNAME=SYS01.DSN3,UNIT=&DEVICE,DISP=&DSPS;
```

From IPCS, issue the following command to consolidate the data from the data sets defined in the cataloged procedure into one data set, GTF.SYS01:

```
COPYTRC TYPE(GTF)
        INDATASET(SYS01.DSN1,SYS01.DSN2,SYS01.DSN3)
        OUTDATASET(GTF.SYS01)
```

Example: Consolidating GTF Output from Multiple Systems

In the following example, the COPYTRC subcommand is used to consolidate data from 3 systems, in data sets GTF.SYS01, GTF.SYS02, and GTF.SYS03, into one output data set, GTF.ALLSYS.

```
COPYTRC TYPE(GTF)
        INDATASET(GTF.SYS01,GTF.SYS02,GTF.SYS03)
        OUTDATASET(GTF.ALLSYS)
```

Note that just one data set per system was used on the COPYTRC command. For best results, if you have more than one data set for a system, you should first consolidate those using a separate instance of the COPYTRC command. See *Example: Consolidating GTF Output from Multiple Data Sets on One System* for more information.

To format the output data set for GTF data, issue the following IPCS subcommand:

```
GTFTRACE DSNAME(ALLSYS)
```

Merging Trace Output

Use the IPCS MERGE subcommand to merge multiple traces into one chronological sequence. The traces can be all of the following:

- Component traces from the same dump on direct access storage (DASD)
- Component traces from different dumps on DASD
- GTF trace records from a dump or data set and on tape or DASD

Example: Merging GTF Output from Multiple Systems

In the following example, the MERGE subcommand is used to consolidate and format data from 3 systems, in data sets GTF.SYS04, GTF.SYS05, and GTF.SYS06, into one chronological sequence in output data set, GTF.SYSALL.

```
MERGE
GTFTRACE DSNAME(GTF.SYS04)
GTFTRACE DSNAME(GTF.SYS05)
GTFTRACE DSNAME(GTF.SYS06)
MERGEEND
```

Reading GTF Output

This topic shows the format of the trace records that GTF creates. When you select your tracing options carefully, GTF provides detailed information about the system and user events where your problem lies, making it easier to diagnose.

This section contains the following topics:

- “Formatted GTF Trace Output” on page 10-35 which has information about trace records formatted by the IPCS GTFTRACE subcommand.
- “Unformatted GTF Trace Output” on page 10-77 which has information about unformatted trace records.

The figure that follows shows the GTF trace options and the trace records they generate in GTF trace output. Use this figure to correlate the options you selected with their associated trace records. Some trace options in the table do not have trace records associated with them:

- ASIDP - Specifies that GTF trace only events from the select address spaces.
- JOBNAMEP - Specifies that GTF trace only events in selected jobs.
- END - Specifies the end of prompting keyword values specified.
- TRC - Specifies that GTF tracing includes the GTF address space.

Trace Options	Trace Record Identifier
ASIDP	N/A
CCW	CCW
CCWP	CCW
CSCH	CSCH
DSP	DSP, LSR, SDSP, SRB
END	N/A
EXT	EXT
HSCH	HSCH
IO	EOS, IO
IOX	IOX
IOP	EOS, IO
IOXP	IOX
JOBNAMEP	N/A
MSCH	MSCH
PCI	PCI
PI	PGM, PI
PIP	PGM, PI
RNIO	RNIO
RR	FRR, STAE
SIO	RSCH, SSCH
SIOP	RSCH, SSCH
SLIP	SLIP
SRM	SRM
SSCH	RSCH, SSCH
SSCHP	RSCH, SSCH
SVC	SVC
SVCP	SVC
SYS	CSCH, EOS, EXT, FRR, HSCH, IO, MSCH, PGM, PI, SSCH, STAE, SVC
SYSM	CSCH, EOS, EXT, FRR, HSCH, IO, MSCH, PGM, PI, SSCH, STAE, SVC
SYSP	CSCH, EOS, EXT, FRR, HSCH, IO, MSCH, PGM, PI, SSCH, STAE, SVC
TRC	N/A
USR	USR
USRP	USR

Figure 10-3. GTF Trace Options and Associated Trace Record Identifiers

Formatted GTF Trace Output

This topic describes GTF trace output records formatted by the IPCS GTFTRACE subcommand. In each formatted record, the length of each field is indicated by the number of characters. The characters indicate the type of data in the field, as follows:

c Character

d Decimal

h Hexadecimal

x Variable information

y Variable information

The CCW trace record format uses additional letters to distinguish parts of fields.

A trace record can contain indicators to denote unusual conditions that occurred while GTF was tracing the event for the record. The indicators are:

N/A
: Not applicable. The field does not apply in this record. In a 2-byte field, not applicable appears as `N/`.

U/A
: Unavailable. GTF could not gather the information. In a 2-byte field, unavailable appears as `U/`.

PPPPPPPP
: Unavailable because of a page fault encountered while GTF was gathering the data (SVC only).

SSSSSSSS
: Unavailable because of security considerations (SVC only).

****
: Unavailable because of an error that occurred while GTF was gathering the data or due to the data being paged out.

X'EEEE'
: Unavailable because of a severe error that occurred while GTF was gathering the data. This value appears in the first 2 data bytes of the trace record. The contents of the trace record are unpredictable.

Trace Record Identifiers

Each trace record has an identifier to indicate the type of record. The following table lists the identifiers alphabetically and gives the page that shows the format for the record.

Trace Record Identifier	GTF Trace Record	Parameter in SYS1.PARMLIB Member or Operator Reply	For format, see topic
****	Time stamp		10-39
****	Lost event		10-40
CCW	Channel program	CCW	10-41
CSCH	Clear subchannel operation	CSCH, SYS, SYSM, SYSP	10-42
DSP	Task dispatch	DSP	10-44
EOS	End-of-sense interruption	IO, IOP, SYS, SYSM, SYSP	10-45
EXT	General external interruption	EXT, SYS, SYSM, SYSP	10-47
FRR	Functional recovery routine return	RR, SYS, SYSM, SYSP	10-49
HEXFORMAT	Unformatted trace event		10-50
HSCH	Halt subchannel operation	HSCH, SYS, SYSM, SYSP	10-42
IO	Input/output interruption	IO, IOP, SYS, SYSM, SYSP	10-45
IOX	Input/output interruption summary record format	IOX, IOXP, SYS, SYSM, SYSP	10-51
LSR	Local supervisor routine dispatch	DSP	10-54
MSCH	Modify subchannel operation	MSCH, SYS, SYSM, SYSP	10-55
PCI	Program-controlled input/output interruption	PCI	10-45
PGM	Program interruption	PI, PIP, SYS, SYSM, SYSP	10-56

Trace Record Identifier	GTF Trace Record	Parameter in SYS1.PARMLIB Member or Operator Reply	For format, see topic
PI	Program interruption	PI, PIP, SYS, SYSM, SYSP	10-56
RNIO	VTAM remote network input/output event	RNIO	10-57
RSCH	Resume subchannel	SSCH, SSCHP	10-58
SDSP	Task re-dispatch	DSP	10-44
SLIP	SLIP program event interruption	SLIP	10-59
SRB	Service request block routine dispatch or re-dispatch	DSP	10-64
SRM	System resources manager return	SRM	10-65
SSCH	Start subchannel operation	SSCH, SSCHP, SYS, SYSM, SYSP	10-66
STAE	STAE or ESTAE recovery routine return	RR, SYS, SYSM, SYSP	10-67
SUBSYS	Unformatted trace event		10-50
SVC	Supervisor call interruption	SVC, SVCP, SYS, SYSM, SYSP	10-69
SYSTEM	Unformatted trace event		10-50
USR	User event	USR, USRP	10-70

Example Formatted GTF Trace output

The following screens show GTF records. IPCS produced the screens from an example dump. These records are in comprehensive format and are time stamped.

The second screen shows records for the start subchannel operation (SSCH) event.

The third IPCS screen shows records for two input/output interruption (IO) events.

The fourth screen shows records for the following events:
- CSCH: clear subchannel operation
- EOS: end-of-sense interruption
- HSCH: halt subchannel operation
- PCI: program-controlled input/output interruption

The subcommand issued on the IPCS Subcommand Entry panel is:

```
GTFTRACE
```

```
IPCS OUTPUT STREAM ------------------------------------------ LINE 0 COLS 1 78
COMMAND ===>                                                  SCROLL ===> CSR

 ****************************** TOP OF DATA ***********************************

   **** GTFTRACE DISPLAY OPTIONS IN EFFECT ****
 SSCH=ALL  IO=ALL  CCW=SI
 SVC=ALL  PI=ALL
 EXT  RNIO  SRM  RR  DSP  SLIP
 **** GTF DATA COLLECTION OPTIONS IN EFFECT: ****
 Minimum tracing for IO, SSCH, SVC, PI, EXT, and FRR events
 All GTRACE events requested
 All events associated with the execution should be traced
 All DISPATCHER events traced
 PCI events are to be traced
 System resource manager events traced

              **** GTF TRACING ENVIRONMENT ****
     Release: SP4.1.0   FMID: HBB4410   System name: FIRST
     CPU Model: 3090  Version: FF  Serial no. 170067

SDSP  ASCB.... 00F49600 CPU..... 0001     PSW..... 070C0000 82200ED2
                        TCB..... 00AF2880 R15..... 00000000 R0...... A9000000
                        R1...... 02200FB0
               GMT-07/02/89 00:29:08.154586    LOC-07/01/89 20:29:08.060378

SVC   CODE.... 047      ASCB.... 00F49600 CPU..... 0001
                        PSW..... 070C002F 82200ED2          TCB..... 00AF2880
                        R15..... 00000000 R0...... A9000000 R1...... 02200FB0
               GMT-07/02/89 00:29:08.154677    LOC-07/01/89 20:29:08.060469
```

```
IPCS OUTPUT STREAM ------------------------------------------ LINE 0 COLS 1 78
COMMAND ===>                                                  SCROLL ===> CSR
SRB   ASCB.... 00F44680 CPU..... 0001     PSW..... 070C0000 82045B48
                        R15..... 82045B48 SRB..... 01AC1520 R1...... 01AC3380
                        TYPE.... INITIAL DISPATCH OF SRB
               GMT-07/02/89 00:29:08.154932    LOC-07/01/89 20:29:08.060724

DSP   ASCB.... 00F44680 CPU..... 0001     PSW..... 070C1000 82360BA2
                        TCB..... 00AF2370 R15..... 80AF2858 R0...... 00000001
                        R1...... FDC9E5D4
               GMT-07/02/89 00:29:08.155169    LOC-07/01/89 20:29:08.060961

 SSCH.... 0223     ASCB.... 00F44680 CPUID... 0001     JOBN.... GTFCBM
                   RST..... 00A4EB00 VST..... 00AF0B00 DSID.... 00AF1A0C
                   CC...... 00       ORB..... 00F684B8 0000E000 00FEC630
                   SEEKA... 00000002 7B000506          GPMSK... 00
                   OPT..... 00       FMSK.... 18       DVRID... 02
                   IOSLVL.. 01       UCBLVL.. 01
               GMT-07/02/89 00:29:08.156738    LOC-07/01/89 20:29:08.062530

DSP   ASCB.... 00FD3300 CPU..... 0001     PSW..... 070E0000 00000000
                        TCB..... 00000000 R15..... ****     R0...... ****
                        R1...... ****
               GMT-07/02/89 00:29:08.157022    LOC-07/01/89 20:29:08.062814
```

```
 IO...... 0223     ASCB.... 00F44680 CPUID... 0001     JOBN.... GTFCBM
                   PSW..... 070E0000 00000000
                   IRB..... 00004007 00A4EB38 0C000000 0040002E
                   TCB..... 00AF2958 SENSE... N/A      FLA..... 00
                   OPT..... 00       DVRID... 02       IOSLVL.. 01
                   UCBLVL.. 01
               GMT-07/02/89 00:29:08.167605    LOC-07/01/89 20:29:08.073397


  **** GTFTRACE DISPLAY OPTIONS IN EFFECT ****
SSCH=ALL  IO=ALL  CCW=SI
SVC=ALL  PI=ALL
EXT  RNIO  SRM  RR  DSP  SLIP
 **** GTF DATA COLLECTION OPTIONS IN EFFECT: ****
IO filtering requested
CCW trace prompting
IO CCW records
SSCH CCW records
All records timestamped
SSCH prompting
*** DATE/TIME: GMT-10/02/90 21:14:10    LOC-10/02/90 21:14:10.009827
IO...... 0000     ASCB.... 00000000 CPUID... 0000     JOBN.... ........
                  PSW..... 00000000 00000000          IRB..... 00000000
                           00000000 00000000 00000000 TCB..... 00000000
                  SENSE... 0000     FLA..... 00       OPT..... 00
                  DVRID... 00       IOSLVL.. 00       UCBLVL.. 00
```

```
              GMT-10/02/90 21:14:10.009803    LOC-10/02/90 21:14:10.009803
PCI..... 0000     ASCB.... 00000000 CPUID... 0000     JOBN.... ........
                  PSW..... 00000000 00000000          IRB..... 00000000
                           00000000 00000000 00000000 TCB..... 00000000
                  SENSE... 0000     FLA..... 00       OPT..... 00
                  DVRID... 00       IOSLVL.. 00       UCBLVL.. 00
              GMT-10/02/90 21:14:10.009955    LOC-10/02/90 21:14:10.009955
EOS..... 0000     ASCB.... ¸GN,     CPUID... 8861     JOBN.... ........
                  PSW..... 00000000 00000000          IRB..... 00000000
                           00000000 00000000 00000000 TCB..... 00000000
                  SENSE... 0000     FLA..... 00       OPT..... 00
                  DVRID... 00       IOSLVL.. 00       UCBLVL.. 00
              GMT-10/02/90 21:14:10.078486    LOC-10/02/90 21:14:10.078486
CSCH.... 0000     ASCB.... 00000000 CPUID... 0000     JOBN.... ........
                  DEV..... 0000     SFLS.... 0000     SID..... 00000000
                  CC...... 00       DVRID... 00       ARDID... 00
                  IOSLVL.. 00       UCBLVL.. 00
              GMT-10/02/90 21:14:10.099752    LOC-10/02/90 21:14:10.099752
HSCH.... 0000     ASCB.... ¸GN,     CPUID... 8861     JOBN.... ........
                  DEV..... 0000     SFLS.... 0000     SID..... 00000000
                  CC...... 00       DVRID... 00       ARDID... 00
                  IOSLVL.. 00       UCBLVL.. 00
              GMT-10/02/90 21:14:10.119803    LOC-10/02/90 21:14:10.119803
 ****************************** END OF DATA ***********************************
```

Formatted Trace Records for Events

Time Stamp Records

Purpose

Time stamp records mark the time an event occurred.

Record Format After Each Trace Record

```
GMT-mm/dd/yy hh:mm:ss:dddddd    LOC-mm/dd/yy hh:mm:ss.dddddd
```

GMT-mm/dd/yy hh:mm:ss
Month/day/year and Greenwich mean time given in hour:minute:second format

LOC-mm/dd/yy hh:mm:ss.dddddd
Month/day/year and local time given in hour:minute:second.microsecond format

Source Index Records

Purpose

Source index records are added when GTF trace records are consolidate using the IPCS COPYTRC subcommand. The records identify the system that produced the GTF trace record.

Record Format After Each Trace Record, if the GTF trace records are consolidated with the IPCS COPYTRC subcommand:

```
SOURCE INDEX: 01
```

The source index record indicates that the GTF trace record was produced by the system with identifier 01. Identifiers include the system name and the trace options in effect for that system. The identifiers are listed at the top of the IPCS report.

Lost Event Records

Purpose

A lost event record indicates that GTF lost the trace records for one or more events because of an error or overflow of the trace buffer.

Record Format When GTF Trace Buffer is Lost due to Error

```
**** ONE TRACE BUFFER LOST TIME hh.mm.ss.dddddd
```

hh.mm.ss.dddddd
The time of day (hour.minute.second.microsecond) when GTF placed the first trace record in the buffer.

The size of the GTF trace buffer is:

- Equal to the blocksize used by GTF when writing the trace data, if GTF is writing the trace records to a data set on a direct access storage device (DASD). The system displays the blocksize in message AHL906I. If the records are to be written to a data set, the system issues message AHL906I after starting GTF.
- 32,760 bytes, if GTF is writing the trace records to a data set on tape.
- 32,768 bytes, if GTF is writing the trace records only into internal trace buffers.

Record Format for Number of Trace Events Lost due to Errors or Trace Buffer Overflow

```
****** LOST EVENTS NUM dddddddddddd LOCAL TIME mm/dd/yyyy hh.mm.ss.nnnnnn ***
```

dddddddddd
The number of lost events

mm/dd/yyyy
The date (in month/day/year format) when GTF placed the first trace record in the current trace buffer.

hh.mm.ss.dddddd
The time of day (hour.minute.second.microsecond) when GTF placed the first trace record in the current trace buffer.

CCW Trace Records

Purpose

A CCW record represents the processing of a channel program.

CCW trace records appear following EOS, IO, PCI, RSCH, or SSCH trace records; they do not appear alone. Any of the formats can appear in any combination in one CCW trace record.

Record Format

```
 CCW CHAIN   FORMAT d          ccc               DEV..... hhhh
             ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
Fhhhhhhh ---CCW-- ---CCW-- dddddddd dddddddd | cccccccc |
Fhhhhhhh ---CCW-- ---CCW-- dddddddd dddddddd | cccccccc |
Fhhhhhhh ---CCW-- ---CCW-- dddddddd dddddddd | cccccccc |
                           dddddddd dddddddd | cccccccc |
                           dddddddd dddddddd | cccccccc |
                           dddddddd dddddddd | cccccccc |
                           dddddddd dddddddd | cccccccc |
                             .
                             --Back half of split data--
                             .
                           dddddddd dddddddd | cccccccc |
                           dddddddd dddddddd | cccc     |
Fhhhhhhh ---CCW-- ---CCW-- dddddddd dddddddd | cccccccc |
                           dddddddd dddddddd | ccc      |
         IDAW     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh
         IOSB     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh
         EWAx     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                           hhhhhhhh hhhhhhhh
```

FORMAT d ccc
Format (d) and type of trace event (ccc): EOS, IO, PCI, RSCH, or SSCH. Format is either zero or 1.

DEV hhhh
Device number from the UCBCHAN field of the UCB.

ASCB hhhhhhhh
Same as the ASCB field in the IO, SSCH, RSCH, PCI, or EOS base record.

CPU hhhh
Same as the CPU ID field in the IO, SSCH, RSCH, PCI, or EOS base record.

JOBN cccccccc
Same as the job named (JOBN) field in the IO, SSCH, RSCH, PCI, or EOS base record.

Fhhhhhhh
Fullword address of the CCW. If the high order bit of the address is on, this is the real address of the CCW, otherwise this is the virtual address of the address of the CCW.

---CCW--
Is the CCW command. The command is either a format 0 or format 1 CCW.
Format 0 CCW is in the format ooaaaaaa ffuubbbb
Format 1 CCW is in the format ooffbbbb aaaaaaaa
where:

oo op code.

aaaaaa
Real address of data associated with the CCW. If indirect address words (IDAWs) are present, this is the address of the IDAW list.

aaaaaaaa
Fullword real address of data associated with the CCW. If IDAWs are present, this is the address of the IDAW list.

ff CCW flags; if this flag is .... .1.. , then this indicates that an IDAW list is present. If this flag is .... ..1., then a suspend of the channel program was requested.

uu Not used by hardware; could contain a nonzero character.

bbbb Byte count.

dddddddd dddddddd | cccccccc |
Information transferred by the CCW. If there is not a series of dashes in this field, then all transferred data is displayed in four byte sections.

--Back half of split data--
Indicates there were more bytes of information transferred than the system programmer specified on the START command. The default value is 20 bytes. The system programmer can specify the number of bytes to be shown. The specified value is halved; for an odd number, the larger section is shown first. The first section of data displayed comes from the beginning of the buffer from which the data was transferred. The last section comes from the end of the buffer.

IDAW hhhhhhhh or hhhhhhhh_hhhhhhhh
Contents of the IDAW, a fullword real address for 31–bit IDAWs or a doubleword real address for 64–bit IDAWs, followed by a halfword specifying the length of the data at that address. The data at the address follows the halfword length.

IOSB hhhhhhhh
Fullword virtual address of the IOSB followed by the contents of the IOSB.

The fullword at offset X'34' of the IOSB points to an error recover procedure work area (EWA), or is zero. The EWA is traced and documented directly below the IOSB and is formatted in the same manner as the IOSB.

EWAx hhhhhhhh
Fullword virtual address of the error recovery procedure work area, followed by the contents of EWA.

CSCH and HSCH Trace Records

Purpose

CSCH and HSCH records represent a clear subchannel operation and a halt subchannel operation.

Record Formats

```
CSCH.... hhhh   ASCB.... hhhhhhhh CPUID... hhhh     JOBN.... cccccccc
                DEV..... hhhh     SFLS.... hhhh     SID..... hhhhhhhh
                CC...... hh       DVRID... hh       ARDID... hh
                IOSLVL.. hh       UCBLVL.. hh
                UCBWGT.. hh       BASE.... hhhh

HSCH.... hhhh   ASCB.... hhhhhhhh CPUID... hhhh     JOBN.... cccccccc
                DEV..... hhhh     SFLS.... hhhh     SID..... hhhhhhhh
                CC...... hh       DVRID... hh       ARDID... hh
                IOSLVL.. hh       UCBLVL.. hh
                UCBWGT.. hh       BASE.... hhhh
```

CSCH hhhh
HSCH hhhh
Device number from the UCBCHAN field of the UCB.

ASCB hhhhhhhh
Address of the ASCB for the address space that started the I/O operation.

CPUID hhhh
Address of the processor on which the I/O operation started

JOBN cccccccc
One of the following:

cccccccc	Name of the job associated with the task that requested the I/O operation
N/A	No job is associated with the requested I/O

DEV hhhh
Device number from the UCBCHAN field of the UCB.

SFLS hhhh
Start flags from the UCBSFLS field of the UCB.

SID hhhhhhhh
Subchannel ID from the UCBSID field of the UCB.

CC hh
CSCH or HSCH condition code in bits 2 - 3.

DVRID hh
Driver ID value from the IOSDVRID field of the IOSB.

ARDID hh
One of the following:

hh	Associated request driver ID from the IOSDVRID field of the IOSB
U/	Unavailable because the IOQ was unavailable

IOSLVL hh
Function level to provide serialization of I/O requests. This value comes from the IOSLEVEL field of the IOSB.

UCBLVL hh
UCB level value from the UCBLEVEL field of the UCB.

UCBWGT hh
Flags from the UCBWGT field of the UCB.

BASE hhhh
Device number from the UCBCHAN field of the UCB (same as DEV hhhh).

DSP and SDSP Trace Records

Purpose

A DSP record represents dispatching of a task. An SDSP record represents re-dispatching of a task after an SVC interruption.

Minimal Trace Record Formats

```
DSP   ASCB.... hhhhhhhh CPU..... hhhh     PSW..... hhhhhhhh hhhhhhhh
                        TCB..... hhhhhhhh R15..... hhhhhhhh R0...... hhhhhhhh
                        R1...... hhhhhhhh

SDSP  ASCB.... hhhhhhhh CPU..... hhhh     PSW..... hhhhhhhh hhhhhhhh
                        TCB..... hhhhhhhh R15..... hhhhhhhh R0...... hhhhhhhh
                        R1...... hhhhhhhh
```

Comprehensive Trace Record Formats

```
 DSP         ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
             DSP-PSW. hhhhhhhh hhhhhhhh          TCB..... hhhhhhhh
             MODN.... yyyyyyyy

 SDSP        ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
             DSP-PSW. hhhhhhhh hhhhhhhh          TCB..... hhhhhhhh
             MODN.... yyyyyyyy
```

ASCB hhhhhhhh
Address of address space control block.

CPU hhhh
Address of processor on which task will be dispatched.

PSW hhhhhhhh hhhhhhhh
DSP-PSW hhhhhhhh hhhhhhhh
Program status word under which the task is dispatched.

JOBN cccccccc
One of the following:

cccccccc	Name of the job associated with the task being dispatched
N/A	The record is for a system or started task
PPPPPPPP	A page fault occurred
********	An internal error occurred

TCB hhhhhhhh
Address of the task control block.

R15 hhhhhhhh
R0 hhhhhhhh
R1 hhhhhhhh
Data that will appear in general registers 15, 0, and 1 when the task is dispatched.

MODN cccccccc
cccccccc is one of the following:

mod_name
The name of a module that will receive control when the task is dispatched.

WAITTCB
Indicates that the system wait task is about to be dispatched.

SVC–T2
Indicates that a type 2 SVC routine that resides in the nucleus is about to be dispatched.

SVC–RES
Indicates that a type 3 SVC routine or the first load module of a type 4 SVC routine is about to be dispatched. The routine is located in the pageable link pack area (PLPA).

SVC–cccc
Indicates that the second or subsequent load module of a type 4 SVC routine is about to be dispatched. The module is located in the fixed or pageable link pack area (LPA). The last four characters of the module name are cccc.

****IRB*****
Indicates that an asynchronous routine with an associated interruption request block (IRB) is about to be dispatched. No module name is available.

***ccccccc**
Indicates that error fetch is in the process of loading an error recovery module. The last seven characters of the module name are ccccccc.

PPPPPPPP
A page fault occurred

****
An internal error occurred

EOS, CS, IO, and PCI Trace Records

Purpose

EOS records represent an end of sense interruption, CS records represent an end of sense interruption for devices containing the concurrent sense facility, IO records represent an input/output (I/O) interruption, and PCI records a program-controlled interruption.

Record Formats

```
EOS..... hhhh   ASCB.... hhhhhhhh CPUID... hhhh     JOBN.... cccccccc
                PSW..... hhhhhhhh hhhhhhhh          IRB..... hhhhhhhh
                         hhhhhhhh hhhhhhhh hhhhhhhh TCB..... hhhhhhhh
                SENSE... hhhh     FLA..... hh       OPT..... hh
                DVRID... hh       IOSLVL.. hh       UCBLVL.. hh
                UCBWGT.. hh       BASE.... hhhh

CS...... hhhh   ASCB.... hhhhhhhh CPUID... hhhh     JOBN.... cccccccc
                PSW..... hhhhhhhh hhhhhhhh          TCB..... hhhhhhhh
                SENSE... hhhh     FLA..... hh       OPT..... hh
                DVRID... hh       IOSLVL.. hh       UCBLVL.. hh
                IRB..... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                         hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                         hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                         hhhhhhhh
                UCBWGT.. hh       BASE.... hhhh
```

```
IO...... hhhh   ASCB.... hhhhhhhh CPUID... hhhh     JOBN.... cccccccc
                PSW..... hhhhhhhh hhhhhhhh          IRB..... hhhhhhhh
                         hhhhhhhh hhhhhhhh hhhhhhhh TCB..... hhhhhhhh
                SENSE... hhhh     FLA..... hh       OPT..... hh
                DVRID... hh       IOSLVL.. hh       UCBLVL.. hh
                UCBWGT.. hh       BASE.... hhhh


PCI..... hhhh   ASCB.... hhhhhhhh CPUID... hhhh     JOBN.... cccccccc
                PSW..... hhhhhhhh hhhhhhhh          IRB..... hhhhhhhh
                         hhhhhhhh hhhhhhhh hhhhhhhh TCB..... hhhhhhhh
                SENSE... hhhh     FLA..... hh       OPT..... hh
                DVRID... hh       IOSLVL.. hh       UCBLVL.. hh
                UCBWGT.. hh       BASE.... hhhh
```

EOS hhhh
CS hhhh
IO hhhh
PCI hhhh
The device number from the UCBCHAN field of the unit control block (UCB).

ASCB {hhhhhhhh|U/A}
One of the following:

hhhhhhhh Address of the address space control block (ASCB) for the address space that started the I/O operation.

U/A Unavailable because the I/O supervisor block (IOSB) control block is unavailable.

CPU hhhh
Address of the processor on which the interruption occurred.

JOBN {cccccccc|N/A|U/A}
One of the following:

cccccccc Name of the job associated with the task that requested the I/O operation.

N/A Not applicable.

U/A Unavailable because the IOSB control block is unavailable.

PSW hhhhhhhh hhhhhhhh
Program status word (PSW) stored when the interruption occurred.

IRB (see explanation)
For the EOS, IO, and PCI trace records, this field contains the first four words, in hexadecimal, of the interruption response block operand of the Test Subchannel (TSCH) instruction. For the CS trace record, this field contains the first 16 words, in hexadecimal, of the interruption response block operand of the TSCH instruction. (Note that this IRB is not the interruption request block indicated as **IRB*** in a DSP trace record.)

TCB {hhhhhhhh|N/A|U/A}
One of the following:

hhhhhhhh Address of the TCB for the task that requested the I/O operation.

N/A Not applicable.

U/A Unavailable because the IOSB control block is unavailable.

SENSE {hhhh|N/A|U/A}
One of the following:

hhhh First 2 sense bytes from the IOSSNS field of the IOSB.

N/A Not applicable.

U/A Unavailable because the IOSB control block is unavailable.

FLA {hh|U/A}
One of the following:

hh	Flag byte from the IOSFLA field of the IOSB.
U/A	Unavailable because the IOSB control block is unavailable.

OPT {hh|U/A}
One of the following:

hh	IOSB options byte from the IOSOPT field of the IOSB.
U/A	Unavailable because the IOSB control block is unavailable.

DVRID {hh|U/A}
One of the following:

hh	Driver identifier from the IOSDVRID field of the IOSB.
U/A	Unavailable because the IOSB control block is unavailable.

IOSLVL {hh|U/A}
One of the following:

hh	Function level to provide serialization of I/O requests. This value comes from the IOSLEVEL field of the IOSB.
U/A	Unavailable because the IOSB control block is unavailable.

UCBLVL hh
UCB level value from the UCBLEVEL field of the UCB.

UCBWGT hh
Flags from the UCBWGT field of the UCB.

BASE hhhh
Device number from the UCBCHAN field of the UCB (same as DEV hhhh).

EXT Trace Records

Purpose

An EXT record represents a general external interruption.

Minimal Trace Record Format

```
EXT  CODE.... hhhh  ASCB.... hhhhhhhh CPU..... hhhh     PSW..... hhhhhhhh
                             hhhhhhhh TCB..... hhhhhhhh ccc-TCB. hhhhhhhh
```

Comprehensive Trace Record Format

```
EXT..... hhhh   ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
                OLD-PSW. hhhhhhhh hhhhhhhh          TCB..... hhhhhhhh
                TQE FIELDS:       FLAGS... hhhh     EXTADDR. hhhhhhhh
                TCB..... hhhhhhhh

EXT..... hhhh   ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
                OLD-PSW. hhhhhhhh hhhhhhhh          TCB..... hhhhhhhh
                TQE FIELDS:       FLAGS... hhhh     EXTADDR. hhhhhhhh
                ASCB.... hhhhhhhh TCB..... hhhhhhhh

EXT..... hhhh   ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
                OLD-PSW. hhhhhhhh hhhhhhhh          TCB..... hhhhhhhh
                PARM.... hhhhhhhh SIG-CPU. hhhh
```

EXT CODE hhhh
EXT hhhh
External interruption code.

ASCB hhhhhhhh
Address of ASCB for the address space that was current when the interruption occurred.

CPU hhhh
Address of the processor on which the interruption occurred.

JOBN cccccccc
One of the following:

cccccccc	Name of the job associated with the interrupted task
N/A	The record is for a system or started task
PPPPPPPP	A page fault occurred
************	An internal error occurred

PSW hhhhhhhh hhhhhhhh
OLD-PSW hhhhhhhh hhhhhhhh
Program status word stored when the interruption occurred.

TCB hhhhhhhh
One of the following:

hhhhhhhh	Address of the TCB for the interrupted task
N/A	Not applicable, as in the case of an interrupted SRB routine

INT-TCB hhhhhhhh
TQE-TCB hhhhhhhh
Address of the TCB. This interruption is indicated by interruption codes 12hh.

TQE FIELDS
Indicates a clock comparator or CPU timer interruption. These interruptions are indicated by interruption codes X'1004' or X'1005'. The following fields contain information from the timer queue element (TQE):

FLAGS hhhh
The flags from the TQEFLGS field.

EXTADDR hhhhhhhh
The first four hexadecimal digits are the contents of the TQEFLGS field; the last four hexadecimal digits are the contents of the TQEEXIT field.

ASCB hhhhhhhh
One of the following:

hhhhhhhh	Contents of the TQEASCB field.
PPPPPPPP	A page fault occurred

**********   An internal error occurred

The TQEASCB field is present only for a clock comparator interruption. TQEASCB contains the address of the ASCB for the address space in which the timer exit routine will be run.

TCB hhhhhhhh
One of the following:

hhhhhhhh Contents of the TQETCB field.
N/A The record is for a system or started task
PPPPPPPP A page fault occurred
********** An internal error occurred

TQETCB contains the address of the TCB for the task under which the timer exit routine will be run.

PARM hhhhhhhh
Signal passed on a signal processor interruption, which is indicated by interruption codes 12hh.

SIG-CPU hhhh
Address of the processor on which a signal processor interruption occurred.

FRR Trace Records

Purpose

An FRR record represents the return to the recovery termination manager (RTM) from a functional recovery routine (FRR). All fields, except the processor address, are gathered from the system diagnostic word area (SDWA) that was passed to the FRR.

Minimal Trace Record Format

```
FRR   ASCB.... hhhhhhhh CPU..... hhhh     PSW..... hhhhhhhh hhhhhhhh
                        CC...... hhhhhhhh FLG1.... hhhhhhhh FLG2.... hhhhhhhh
                        RETRY... hhhhhhhh RTCA.... hhhhhhhh
```

Comprehensive Trace Record Format

```
 FRR           ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
               NAME.... cccccccc PSW..... hhhhhhhh hhhhhhhh
               ABCC.... hhhhhhhh ERRT.... hhhhhhhh FLG..... hhhhhh
               RC...... hh       RTRY.... hhhhhhhh
```

ASCB hhhhhhhh
One of the following:

hhhhhhhh Address of the ASCB for the address space in which the error occurred.
PPPPPPPP A page fault occurred
********** An internal error occurred

CPU hhhh
Address of the processor associated with the error.

JOBN cccccccc
One of the following:

cccccccc Name of the job associated with the error
N/A The record is for a system or started task
PPPPPPPP A page fault occurred

************ An internal error occurred

NAME cccccccc
Name of the FRR routine.

PSW hhhhhhhh hhhhhhhh
One of the following:

hhhhhhhh hhhhhhhh
Program status word at the time of the error

PPPPPPPP A page fault occurred

************ An internal error occurred

CC hhhhhhhh
ABCC hhhhhhhh
One of the following:

hhhhhhhh The first three digits are the system completion code and the last three digits are the user completion code

U/A Unavailable because the system diagnostic work area (SDWA) was unavailable

************ An internal error occurred

FLG1 hhhhhhhh
FLG hhhhhh
ERRT hhhhhhhh
Error-type flags from the SDWAFLGS field of SDWA.

FLG2 hhhhhh
Additional flags from the SDWAMCHD and SDWAACF2 fields of SDWA. The flags are contained in the two low-order bytes of this printed field; the high order byte is meaningless.

RC hh
Return code

RETRY hhhhhhhh
RTRY hhhhhhhh
One of the following:

hhhhhhhh Retry address supplied by the FRR

N/A Not applicable, indicating an FRR return code other than 4

PPPPPPPP A page fault occurred

************ An internal error occurred

RTCA hhhhhhhh
Indicates if the recovery routine was a STAE or ESTAE.

HEXFORMAT, SUBSYS, and SYSTEM Trace Records

Purpose

HEXFORMAT, SUBSYS, and SYSTEM records represent events for which GTF could not format the records.

Record Formats

```
HEXFORMAT   AID hh FID hh EID hh hhhhhhhh hhhhhhh ...

SUBSYS      AID hh FID hh EID hh hhhhhhhh hhhhhhh ...

SYSTEM      AID hh FID hh EID hh hhhhhhhh hhhhhhh ...
```

HEXFORMAT
Indicates an event signalled by a GTRACE macro. The macro specified no formatting routine (FID=00).

SUBSYS
Indicates an event signalled by a GTRACE macro. The macro specified a formatting routine (FID=hh) that could not be found.

SYSTEM
Indicates a system event. The trace record could not be formatted for one of the following reasons:

- If EEEE hex appears in bytes 0-1 or 8-9 of the recorded data, an unrecoverable error occurred in a GTF data-gathering routine. Message AHL118I is written on the console, identifying the module that caused the error and the action taken. (The message indicates that GTF will no longer trace this type of event. No more records for this type of event will appear in the trace output.)
- If EEEE hex does not appear in bytes 0-1 or 8-9 of the recorded data, the record could not be formatted because the GTF formatting routine could not be found.

AID hh
Application identifier, which should always be AID FF.

FID hh
Format identifier of the routine (AMDUSRhh or AMDSYShh) that was to format this record.

EID hh
Event identifier, which uniquely identifies the event that produced the record.

hhhhhhhh hhhhhhhh ...
Recorded data (256 bytes maximum).

IOX Trace Records

Purpose

IOX records represent an input/output (I/O) interruption for a completed channel program and a summary of a complete channel program for the I/O operation.

Record Formats

```
IOX....hhhh ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
            DEVN.... hhhh     SID..... hhhh     DRID.... hh
            TVSN.... hh       ECNO.... hhhh     DVCLS... hh
            DSTAT... hh       AERRC... hh       FLAG0... hh
            VOLSER.. cccccc   UCBTYP.. hhhhhhhh
            DSNAME.. cccc.cccccc.cccccc.ccccc
            NSSCH... hhhh     DSSCH... hhhh     SDCON... hhhhhhhh
            SRPEN... hhhhhhhh SDISC... hhhhhhhh SCUQU... hhhhhhhh
            IODTS... hhhhhhhh hhhhhhhh
            AONLY... hhhhhhhh

CCW SECTION   SQNO.... hh       FGS1.... hh       FGS2.... hh
              RCNT.... hh       BLKR.... hhhh     BLKW.... hhhh
              BTRD.... hhhhhhhh BTWR.... hhhhhhhh DCHN.... hhhh
              CCHN.... hhhh     DEGA.... hh       DEGE.... hh
              DEEE.... hhhhhhhh SEEKLOCC hh       CCHHR... hhhhhhhh hh
```

IOX hhhh
IOX identifies the beginning of an IOX record where hhhh is the device number.

ASCB {hhhhhhhh|U/A}
One of the following:

hhhhhhhh Address of the address space control block (ACSB) for the address space that started the I/O operation.

U/A Unavailable because the I/O supervisor block (IOSB) control block is unavailable.

CPU hhhh
Address of the processor on which the interruption occurred.

JOBN {cccccccc|N/A|U/A}
One of the following:

cccccccc Name of the job associated with the task that requested the I/O operation.

N/A Not applicable.

U/A Unavailable because the IOSB control block is unavailable.

DEVN hhhh
Device number

SID hhhh
System ID

DRID hh
Driver ID from IOSB

TVSN hh
Trace version

ECNO hhhh
Record count

DVCLS hh
Device class

DSTAT hh
Device status

AERRC hh
Error codes found during CCW analysis

FLAG0 hh
Flag byte

VOLSER ccccccc
Volume Serial

UCBTYP hhhhhhhh
UCB type

DSNAME cccc.cccccc.cccccc.ccccc
44-byte data set name

NSSCH hhhh
Number of SSCH instructions.

DSSCH hhhh
Number of SSCH instructions for which data was collected.

SDCON hhhhhhhh
Summation of device connect times.

SRPEN hhhhhhhh
Summation of SSCH request pending times.

SDISC hhhhhhhh
Summation of subchannel disconnect times.

SCUQU hhhhhhhh
Summation of control unit queuing times

IODTS hhhhhhhh
Time stamp from IOD

AONLY hhhhhhhh
Device active only time

SQNO hh
Orientation Sequence Number

FGS1 hh
Flag byte 1

FGS2 hh
Flag byte 2

RCNT hh
Count of erase

BLKR hhhh
Number of blocks read

BLKW hhhh
Number of block written

BTRD hhhhhhhh
Number of bytes read

BTWR hhhhhhhh
Number of bytes written

DCHN hhhh
Number of data chain CCWs

CCHN hhhh
Number of COM chain CCWs

DEGA hh
Definition of exterior global attribute

DEGE hh
Definition of exterior global attribute extended

DEEE hhhhhhh
Definition of exterior end of extend CCH

SEEKLOCC hh
Seek/locate code

CCHHR hhhhhhhh
CCHHR seek or search address

LSR Trace Records

Purpose

An LSR record represents dispatching of a local supervisor routine in an address space.

Minimal Trace Record Format

```
LSR   ASCB.... hhhhhhhh CPU..... hhhh     PSW..... hhhhhhhh hhhhhhhh
                        TCB..... hhhhhhhh R15..... hhhhhhhh R0...... hhhhhhhh
                        R1...... hhhhhhhh
```

Comprehensive Trace Record Format

```
 LSR          ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
              LSR-PSW. hhhhhhhh hhhhhhhh          TCB..... hhhhhhhh
```

ASCB hhhhhhhh
Address of the address space control block.

CPU hhhh
Address of the processor on which the routine will be dispatched.

PSW hhhhhhhh hhhhhhhh
LSR-PSW hhhhhhhh hhhhhhhh
Program status word under which the routine will receive control.

JOBN cccccccc
One of the following:

cccccccc	Name of the job associated with the routine being dispatched
N/A	Not applicable
PPPPPPPP	A page fault occurred
**********	An internal error occurred

TCB hhhhhhhh
One of the following:

hhhhhhhh	Address of the task control block associated with this routine (if the routine is run as part of a task)
N/A	Not applicable

R15 hhhhhhhh
R0 hhhhhhhh
R1 hhhhhhhh
One of the following:

hhhhhhhh	Data that will appear in general registers 15, 0, and 1 when the local supervisor routine is dispatched

PPPPPPPP A page fault occurred
************ An internal error occurred

MSCH Trace Records

Purpose

An MSCH record represents a modify subchannel operation.

Record Format

```
MSCH.... hhhh  ASCB.... hhhhhhhh CPUID... hhhh     JOBN.... cccccccc
               SID..... hhhhhhhh CC...... hh       OPT..... hh
               OPT2.... hh       IOSLVL.. hh       SCHIB1.. hhhhhhhh
                        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                        hhhhhhhh UCBLVL.. hh       SCHIB2.. hhhhhhhh
                        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                        hhhhhhhh
                        hhhhhhhh UCBWGT.. hh       BASE.... hhhh
```

MSCH hhhh
Device number from the UCBCHAN field of the UCB.

ASCB hhhhhhhh
Address of the ASCB for the address space that started the modify subchannel operation.

CPU hhhh
Address of the processor on which the modify subchannel started.

JOBN cccccccc
One of the following:

cccccccc Name of the job associated with the task that requested the modify subchannel operation
N/A Not applicable

SID hhhhhhhh
Subchannel ID from the UCBSID field of the UCB.

CC hh
MSCH condition code in bits 2 and 3.

OPT hh
IOSB option bytes from the IOSOPT field of the IOSB.

OPT2 hh
IOSB option bytes from the IOSOPT field of the IOSB.

IOSLVL hh
Function level to provide serialization of I/O requests. This value comes from the IOSLEVEL field of the IOSB.

SCHIB1 hhhhhhhh ... hhhhhhhh
First 7 words of the subchannel information block. Input from the caller of modify subchannel instruction. SCHIB address from the IOSSCHIB field of the IOSB.

UCBLVL hh
UCB level value from the UCBLEVEL field of the UCB.

SCHIB2 hhhhhhhh ... hhhhhhhh
First 7 words of the subchannel information block resulting from the modify subchannel instruction.

UCBWGT hh
Flags from the UCBWGT field of the UCB.

BASE hhhh
Device number from the UCBCHAN field of the UCB (same as DEV hhhh).

PGM and PI Trace Records

Purpose

PGM and PI records represent program interruptions.

Minimal Trace Record Format

```
PI    CODE.... hhh    ASCB.... hhhhhhhh CPU..... hhhh     PSW..... hhhhhhhh
                               hhhhhhhh TCB..... hhhhhhhh VPH..... hhhhhhhh
                      VPA..... hhhhhhhh R15..... hhhhhhhh R1...... hhhhhhhh
```

Comprehensive Trace Record Format

```
PGM..... hhh   ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
               OLD-PSW. hhhhhhhh hhhhhhhh          TCB..... hhhhhhhh
               VPH..... hhhhhhhh VPA..... hhhhhhhh MODN.... cccccccc
               R0...... hhhhhhhh R1...... hhhhhhhh R2...... hhhhhhhh
               R3...... hhhhhhhh R4...... hhhhhhhh R5...... hhhhhhhh
               R6...... hhhhhhhh R7...... hhhhhhhh R8...... hhhhhhhh
               R9...... hhhhhhhh R10..... hhhhhhhh R11..... hhhhhhhh
               R12..... hhhhhhhh R13..... hhhhhhhh R14..... hhhhhhhh
               R15..... hhhhhhhh
```

PI CODE hhh
PGM hhh
Program interruption code, in decimal.

ASCB hhhhhhhh
Address of ASCB for the address space in which the interruption occurred.

CPU hhhh
Address of the processor on which the interruption occurred.

JOBN cccccccc
One of the following:

cccccccc	Name of the job associated with the interruption
N/A	Not applicable
PPPPPPPP	A page fault occurred
************	An internal error occurred

PSW hhhhhhhh hhhhhhhh
OLD-PSW hhhhhhhh hhhhhhhh
Program status word stored when the interruption occurred.

TCB hhhhhhhh
One of the following:

hhhhhhhh	Address of the TCB for the interrupted task
N/A	Not applicable as in the case of an interrupted SRB routine

VPH hhhhhhhh
VPA hhhhhhhh
Virtual page address high half, in the case of a 64–bit translation exception address (TEA) value greater than X'FFFFFFFF' is stored. Virtual page address, in the case of a translation process exception resulting from a reference to the page. This area is meaningless for other types of program interruptions.

MODN cccccccc
cccccccc is one of the following:

mod_name
The name of a module that will receive control when the task is dispatched.

WAITTCB
Indicates that the system wait task was interrupted.

SVC-T2
Indicates that a type 2 SVC routine resident in the nucleus was interrupted.

SVC-RES
Indicates that a type 2 SVC routine or the first load module of a type 4 SVC routine was interrupted. The routine is located in the pageable link pack area (PLPA).

SVC-ccc
Indicates that the second or subsequent load module of a type 4 SVC routine was interrupted. The module is located in the fixed or pageable link pack area (LPA). The last four characters of the load module name are cccc.

****IRB*****
Indicates that an asynchronous routine with an associated interrupt request block was interrupted. No module name is available.

***ccccccc**
Indicates that an error recovery module was in control. The last seven characters of the module name are ccccccc.

****
An internal error occurred

Rdd hhhhhhhh
Contents of general registers when the interruption occurred.

RNIO Trace Records

Purpose

An RNIO record represents a VTAM remote network input/output event. For trace information, see *z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures*.

Minimal Trace Record Format

```
RNIO  ASCB.... hhhhhhhh CPU..... hhhh     R0...... hhhhhhhh
```

Comprehensive Trace Record Format

```
 RNIO          ASCB.... hhhhhhhh CPU..... hhhh     JOB..... cccccccc
               IN...... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
               R0...... hhhhhhhh

 RNIO          ASCB.... hhhhhhhh CPU..... hhhh     JOB..... cccccccc
               OUT..... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
               R0...... hhhhhhhh
```

ASCB hhhhhhhh
Address of the ASCB for the address space of the application associated with the event.

CPU hhhh
Address of the processor that ran the I/O instruction.

JOBN cccccccc
One of the following:

cccccccc	Name of the job associated with the IO event
N/A	Not applicable
PPPPPPPP	A page fault occurred
************	An internal error occurred

IN hhhhhhhh ... hhhhhhhh
OUT hhhhhhhh ... hhhhhhhh
IN indicates that the I/O is from NCP to VTAM; OUT indicates that the direction of the I/O is from VTAM to NCP. The hexadecimal data is:

- For IN events: the transmission header, the response header, and the response unit.
- For OUT events: the transmission header, the request header, and the request unit.

R0 hhhhhhhh
Contents of general register 1 when the event occurred.

RSCH Trace Records

Purpose

An RSCH record represents a resume subchannel operation.

Record Format

```
RSCH.... hhhh  ASCB.... hhhhhhhh CPUID... hhhh     JOBN.... cccccccc
               RST..... hhhhhhhh VST..... hhhhhhhh DSID.... hhhhhhhh
               CC...... hh       SEEKA... hhhhhhhh hhhhhhhh
               GPMSK... hh       OPT..... hh       FMSK.... hh
               DVRID... hh       IOSLVL.. hh       UCBLVL.. hh
               UCBWGT.. hh       BASE.... hhhh
```

RSCH hhhh
Device number from the UCBCHAN field of the UCB.

ASCB hhhhhhhh
Address of the ASCB for the address space that started the I/O operation.

CPU hhhh
Address of the processor on which the I/O operation resumed.

JOBN cccccccc
One of the following:

cccccccc	Name of the job associated with the I/O operation
N/A	Not applicable

RST hhhhhhhh
Address of the channel program. This value comes from the contents of the IOSRST field of the IOSB.

VST hhhhhhhh
Virtual address of the channel program. This value comes from the contents of the IOSVST field of the IOSB.

DSID hhhhhhhh
Request identifier used by purge. Contents of the IOSDID field of the IOSB (address of the DEB or another control block used by purge).

CC hh
RSCH condition code in bits 2 and 3.

SEEKA hhhhhhhh hhhhhhhh
Dynamic seek address from the IOSEEKA field of the IOSB.

GPMSK hh
Guaranteed device path mask for GDP requests from the IOSEEKA field of the IOSB.

OPT hh
IOSB options byte from the IOSOPT field of the IOSB.

FMSK hh
Mode set/file mask from the IOSFMSK field of the IOSB.

DVRID hh
Driver ID from the IOSDVRID field of the IOSB.

IOSLVL hh
Function level to provide serialization of I/O requests. This value comes from the IOSLEVEL field of the IOSB.

UCBLVL hh
UCB level value from the UCBLEVEL field of the UCB.

UCBWGT hh
Flags from the UCBWGT field of the UCB.

BASE hhhh
Device number from the UCBCHAN field of the UCB (same as DEV hhhh).

SLIP Trace Records

Purpose

A SLIP record represents a SLIP program event interruption. GTF writes four types of SLIP records:
- SLIP standard trace record
- SLIP stand/user trace record
- SLIP user trace record
- SLIP debug trace record

SLIP Standard Trace Record

Purpose

A SLIP standard (STD) trace record represents a slip trap match when the SLIP command specifies ACTION=TRACE or ACTION=TRDUMP.

Record Format

```
SLIP STD    ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
            TID..... cccc     ASID.... hhhh     JSP..... cccccccc
            TCB..... hhhhhhhh MFLG.... hhhh     EFLG.... hhhh
            SFLG.... hh       DAUN.... hhhh     MODN.... cccccccc
            OFFS.... hhhhhhhh IADR.... hhhhhhhh INS..... hhhhhhhh     hhhh
            EXSIAD.. hhhhhhhh EXSINS.. hhhhhhhh BRNGA... hhhhhhhh
            BRNGH... hhhhhhhh BRNGD... hhhhhhhh
            OPSW.... hhhhhhhh hhhhhhhh          PIC/ILC. hhhhhhhh
            PERC.... hh       TYP..... hh       PKM..... hhhh
            SASID... hhhh     AX...... hhhh     PASID... hhhh
            ASC..... c        SA-SPACE ccccccccc     cccc
```

ASCB hhhhhhhh
The address of the ASCB for the current address space.

CPU hhhh
The processor identifier (ID).

JOBN cccccccc
One of the following:

cccccccc	Name of the job associated with the SLIP trap
N/A	Not applicable

TID cccc
The trap ID.

ASID hhhh
The identifier of the current address space.

JSP cccccccc
One of the following:

cccccccc	Job step program name
N/A	Not applicable
U/A	Unavailable

TCB hhhhhhhh
One of the following:

hhhhhhhh	TCB address
N/A	Not applicable

MFLG hhhh
System mode indicators that indicate the status of the system. The indicators correspond to the SLWACW field in the SLCA. See *z/OS MVS Data Areas, Vol 4 (RD-SRRA)* for a description of the SLWA.

EFLG hhhh
Error bytes that indicate the error status of the system. These bytes correspond to SDWAERRA in the SDWA. For a description of the SDWA, see *z/OS MVS Data Areas, Vol 4 (RD-SRRA)*.

SFLD hh
SLIP status flags.

DAUN hhhhhhhh
A counter representing the number of times data was unavailable for the DATA keyword test.

The following fields apply to PER interruptions only. For other than PER interruptions, these fields are not applicable and contain: N/A, N/, or N.

MODN cccccccc
One of the following:

cccccccc	Load module name in which the interruption occurred
N/A	Not applicable
U/A	Unavailable

OFFS hhhhhhhh
One of the following:

hhhhhhhh	Offset into the load module containing the instruction that caused the interruption
N/A	Not applicable
U/A	Unavailable

IADR hhhhhhhh
Address of the instruction that caused the interruption.

INS hhhhhhhhhhhh
Instruction content: the instruction that caused the PER interruption.

EXSIAD hhhhhhhh
One of the following:

hhhhhhhh	Target instruction address if the INS field is an Execute instruction
N/A	Not applicable
U/A	Unavailable

EXSINS hhhhhhhhhh
One of the following:

hhhhhhhh	Target instruction content if an INS field is an Execute instruction: 6 bytes of data beginning at the target instruction address
N/A	Not applicable
U/A	Unavailable

BRNGA hhhhhhhh
BRNGH hhhhhhhh
One of the following:

hhhhhhhh	The beginning range virtual address if the SLIP command specified SA
N/A	Not applicable

BRNGD hhhhhhhh
One of the following:

hhhhhhhh	Four bytes of storage starting at the beginning range virtual address if SA was specified
N/A	Not applicable
U/A	Unavailable

OPSW hhhhhhhh hhhhhhhh
The program old PSW.

PIC/ILC hhhhhhhh
The program interruption code and instruction length code.

PERC hh
The PER interruption code.

TYP hh
The PER trap mode.

PKM hhhh
The PSW key mask.

SASID hhhh
The identifier of the secondary address space.

AX hhhh
The authorization index.

PASID hhhh
The identifier of the primary address space.

ASC c
The PSW ASC mode indicator:

c	**Meaning**
0	Primary addressing mode
1	Access register addressing mode
2	Secondary addressing mode
3	Home addressing mode

SA-SPACE cccccccccccc
Storage alteration space identifier, as follows:
- The ASID, for an address space
- The owning ASID and the data space name, for a data space

SLIP Standard/User Trace Record

Purpose

The SLIP standard/user trace record represents a slip trap match when the SLIP command specifies ACTION=TRACE or ACTION=TRDUMP and TRDATA=parameters.

Record Format

```
SLIP S+U   ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
           TID..... cccc     ASID.... hhhh     JSP..... cccccccc
           TCB..... hhhhhhhh MFLG.... hhhh     EFLG.... hhhh
           SFLG.... hh       DAUN.... hhhh     MODN.... cccccccc
           OFFS.... hhhhhhhh IADR.... hhhhhhhh INS..... hhhhhhhh         hhhh
           EXSIAD.. hhhhhhhh EXSINS.. hhhhhhhh BRNGA... hhhhhhhh
           BRNGD... hhhhhhhh OPSW.... hhhhhhhh hhhhhhhh
           PIC/ILC. hhhhhhhh PERC.... hh       TYP..... hh
           PKM..... hhhh     SASID... hhhh     AX...... hhhh
           PASID... hhhh     ASC..... c        SA-SPACE ccccccccc        cccc
           GENERAL PURPOSE REGISTER VALUES
           0-3..... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           4-7..... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           8-11.... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           12-15... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           GPR HIGH HALF VALUES
           0-3..... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           4-7..... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           8-11.... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           12-15... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
            ACCESS REGISTER VALUES
           0-3..... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           4-7..... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           8-11.... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
           12-15... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
```

ASCB hhhhhhhh . . . SA-SPACE cccccccccccc
These fields are the same as the fields in the SLIP standard trace record.

GENERAL PURPOSE REGISTER VALUES
GPR HIGH HALF VALUES

ACCESS REGISTER VALUES
Contents of the general purpose registers and access registers at the time of the error or interruption, if REGS is specified in TRDATA on the SLIP command. The GPR high half values will only be traced in z/Architecture mode.

SLIP User Trace Record

Purpose

The SLIP user record represents a SLIP trap match when the SLIP command specifies ACTION=TRACE or ACTION=TRDUMP and TRDATA=parameters.

Record Format

```
SLIP USR    CPU..... hhhh     EXT..... hhhh     CNTLN... hh
      hhhh  hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh | cccccccccccccccc |
```

CPU hhhh
Processor ID.

EXT hhhh
Extension number.

CNTLN hh
Continuation length.

hhhh
Length for the single range in the SLIP command. If hhhh is zero, either the range was not available or the range was not valid, so that GTF did not collect data for the range. GTF would consider the range not valid if, for example, the ending range address precedes the beginning range address.

hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh | cccccccccccccccc |
User-defined data fields that are specified by TRDATA on the SLIP command.

The length and data fields may be repeated.

For a SLIP command, the trace contains as many user records and user continuation records as needed to trace the data ranges specified in the TRDATA parameter on the SLIP command. The header in each record contains the processor ID and the extension number. When a record is filled enough so that the next data range cannot fit, GTF writes the partially filled record to the GTF trace table. GTF builds another record; its extension number is increased by one and the continuation length is set to zero.

When the length of data from a range is greater than 249 bytes, the excess data is put in user continuation records. After writing the SLIP USR record, GTF builds a user continuation record. GTF increases the extension number by one and sets the continuation length to the number of bytes of data to be put in the continuation record. If more than 251 bytes of data are left, GTF copies 248 bytes into the record and places it in the GTF trace table. GTF builds user continuation records until all the data from a range is traced.

SLIP Debug Trace Record

Purpose

The SLIP debug record represents a SLIP trap match when the SLIP command specifies DEBUG.

Record Format

```
SLIP S+U     ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
             TID..... cccc     ASID.... hhhh     JSP..... cccccccc
             TCB..... hhhhhhhh MFLG.... hhhh     EFLG.... hhhh
             SFLG.... hh       DAUN.... hhhh     MODN.... cccccccc
             OFFS.... hhhhhhhh IADR.... hhhhhhhh INS..... hhhhhhhh       hhhh
             EXSIAD.. hhhhhhhh EXSINS.. hhhhhhhh BRNGA... hhhhhhhh
             BRNGD... hhhhhhhh OPSW.... hhhhhhhh hhhhhhhh
             PIC/ILC. hhhhhhhh PERC.... hh       TYP..... hh
             hh00
```

ASCB hhhhhhhh . . . TYP hh
These fields are the same as the fields in the SLIP standard trace record. The high order bit in the SFLG field is set to 1 to indicate a debug record.

hh00
Two bytes of debug-produced data. The first byte indicates which keyword failed, the second byte contains zeros.

Byte 1 (decimal)	Keyword That Failed
...1	DATA test failed
3	ASID
4	JOBNAME
5	JSPGM
6	PVTMOD
7	LPAMOD
8	ADDRESS
9	MODE
..10	ERRTYP
13	RANGE
14	DATA
20	ASIDSA
22	REASON CODE
23	NUCMOD
24	PSWASC
26	DSSA

SRB Trace Records

Purpose

An SRB record represents dispatching of an asynchronous routine represented by a service request block (SRB).

Minimal Trace Record Format

```
SRB   ASCB.... hhhhhhhh CPU..... hhhh     PSW..... hhhhhhhh hhhhhhhh
                        R15..... hhhhhhhh SRB..... hhhhhhhh R1...... hhhhhhhh
                        TYPE.... ccccccccccccccccccccccccccc
```

Comprehensive Trace Record Format

```
SRB          ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
             SRB-PSW. hhhhhhhh hhhhhhhh          SRB..... hhhhhhhh
                        TYPE.... ccccccccccccccccccccccccccc
```

ASCB hhhhhhhh
Address of the ASCB for the address space in which the SRB routine is dispatched. This may or may not be the address space in which the SRB was created.

CPU hhh
Address of the processor on which the SRB routine is dispatched.

PSW hhhhhhhh hhhhhhhh
SRB-PSW hhhhhhhh hhhhhhhh
Program status word under which the SRB routine receives control.

JOBN cccccccc
One of the following:

cccccccc	Name of the job associated with the SRB being dispatched
N/A	Not applicable, as in the case of a global SRB, which is indicated in the TYPE field
************	An internal error occurred

SRB hhhhhhhh
One of the following:

hhhhhhhh	Address of the service request block (SRB)
************	An internal error occurred

R15 hhhhhhhh
R1hhhhhhhh
One of the following:

hhhhhhhh	Data that will appear in general registers 15 and 1 when the SRB routine is dispatched
************	An internal error occurred

PARM hhhhhhhh
One of the following:

hhhhhhhh	Four-byte parameter or the address of a parameter field to be passed to the SRB routine
N/A	Not applicable, as in the case of a suspended SRB, which is indicated in the TYPE field

TYPE cccccccccccccccccccccccccc
Indicates the type of SRB routine, as follows:

SUSPENDED
Denotes an SRB routine that was dispatched earlier and was subsequently interrupted (for example, by I/O operations or by a request for a lock). The routine is about to be re-dispatched.

INITIAL DISPATCH OF SRB
Denotes an SRB routine selected from the service priority list that is about to be dispatched for the first time.

REDISPATCH OF SUSPENDED SRB
Denotes an SRB routine that was dispatched earlier and was subsequently interrupted (for example, by I/O operations or by a request for a lock). The routine is about to be re-dispatched.

SRM Trace Records

Purpose

An SRM record represents an entry to the system resources manager (SRM).

Minimal Trace Record Format

```
SRM   ASCB.... hhhhhhhh CPU..... hhhh   R15..... hhhhhhhh R0...... hhhhhhhh
                        R1...... hhhhhhhh
```

Comprehensive Trace Record Format

```
SRM          ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
             R15..... hhhhhhhh R0...... hhhhhhhh R1...... hhhhhhhh
```

ASCB hhhhhhhh
One of the following:

hhhhhhhh Address of the ASCB for the address space that was current when SRM was entered

************ An internal error occurred

CPU hhhh
Address of the processor used by the system resources manager.

JOBN cccccccc
One of the following:

cccccccc Name of the job associated with the entry to SRM

N/A Not applicable

************ An internal error occurred

R15 hhhhhhhh
R0 hhhhhhhh
R1 hhhhhhhh
Data that was contained in general registers 15, 0, and 1 when the system resources manager passed control to GTF. The data includes the SYSEVENT code in the low-order byte of register 0.

SSCH Trace Records

Purpose

An SSCH record represents a start subchannel operation.

Record Format

```
SSCH.... hhhh  ASCB.... hhhhhhhh CPUID... hhhh     JOBN.... cccccccc
               RST..... hhhhhhhh VST..... hhhhhhhh DSID.... hhhhhhhh
               CC...... hh       SEEKA... hhhhhhhh hhhhhhhh
               GPMSK... hh       OPT..... hh       FMSK.... hh
               DVRID... hh       IOSLVL.. hh       UCBLVL.. hh
               UCBWGT.. hh       BASE.... hhhh
               ORB..... hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
                        hhhhhhhh hhhhhhhh hhhhhhhh
```

SSCH hhhh
Device number from the UCBCHAN field of the UCB.

ASCB hhhhhhhh
Address of the ASCB for the address space that started the I/O operation.

CPU hhhh
Address of the processor on which the I/O operation started.

JOBN cccccccc
One of the following:

cccccccc Name of the job associated with I/O operation

N/A Not applicable

RST hhhhhhhh
Address of the channel program. This value comes from the contents of the IOSRST field of the IOSB.

VST hhhhhhhh
Virtual address of the channel program. This value comes from the contents of the IOSVST field of the IOSB.

DSID hhhhhhhh
Request identifier used by purge. This identifier is in the IOSDID field of the IOSB and is the address of the DEB or another control block used by PURGE.

CC hh
SSCH condition code in bits 2 and 3.

SEEKA hhhhhhhh hhhhhhhh
Dynamic seek address from the IOSEEKA field of the IOSB.

GPMSK hh
Guaranteed device path mask for GDP requests from the IOSGPMSK field of the IOSB.

OPT hh
IOSB options byte from the IOSOPT field of the IOSB.

FMSK hh
Mode Set/File mask from the IOSFMSK field of the IOSB.

DVRID hh
Driver ID from the IOSDVRID field of the IOSB.

IOSLVL hh
Function level to provide serialization of I/O requests. This value comes from the IOSLEVEL field of the IOSB.

UCBLVL hh
UCB level value from the UCBLEVEL field of the UCB.

UCBWGT hh
Flags from the UCBWGT field of the UCB.

BASE hhhh
Device number from the UCBCHAN field of the UCB (same as DEV hhhh).

ORB hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
Contents of the operation request block (ORB).

STAE Trace Records

Purpose

A STAE record represents return to the recovery termination manager (RTM) from a STAE or ESTAE routine.

Minimal Trace Record Format

```
STAE  PSW..... hhhhhhhh hhhhhhhh  CC...... hhhhhhhh RF...... hhhhhhhh
                        TYCA.... hhhhhhhh hhhhhhhh
```

Comprehensive Trace Record Format

```
STAE        ASCB.... hhhhhhhh CPU..... hhhh     JOBN.... cccccccc
            ESTN.... cccccccc ERR-PSW. hhhhhhhh hhhhhhhh
            ABCC.... hhhhhhhh ERRT.... hhhhhhhh FLG..... hhhhhh
            RC...... hh       RTRY.... hhhhhhhh RTCA.... hhhhhhhh
```

ASCB hhhhhhhh
Address of the ASCB for the address space involved in the recovery.

CPU hhhh
Address of the processor.

JOBN cccccccc
One of the following:

cccccccc	Name of the job involved in the recovery
N/A	Not applicable
************	An internal error occurred

ESTN cccccccc
One of the following:

cccccccc	ESTAE routine name
U/A	Unavailable because the routine did not supply a name
************	An internal error occurred

PSW hhhhhhhh hhhhhhhh
ERR-PSW hhhhhhhh hhhhhhhh
One of the following:

hhhhhhhh hhhhhhhh	Program status word at the time of the error
U/A	Unavailable because the system diagnostic work area (SDWA) was unavailable
************	An internal error occurred

CC hhhhhhhh
ABCC hhhhhhhh
One of the following:

hhhhhhhh	The first four digits are the system completion code and the last four digits are the user completion code
U/A	Unavailable because the system diagnostic work area (SDWA) was unavailable
************	An internal error occurred

TYCA hhhhhhhh hhhhhhhh
Retry address (see RTRY hhhhhhhh following) and an indication of whether the routine was a STAE or ESTAE (see RTCA hhhhhhhh following).

RF hhhhhhhh
FLG hhhhhh
ERRT hhhhhhhh
Error flags from the SDWAFLGS field of the SDWA.

RC hh
Return code

RTRY hhhhhhhh
One of the following:

hhhhhhhh	The address supplied by the FRR
N/A	Not applicable, indicating an FRR return code other than 4
PPPPPPPP	A page fault occurred
************	An internal error occurred

RTCA hhhhhhhh
Indicates if the recovery routine was a STAE or ESTAE.

SVC Trace Records

Purpose

An SVC record represents a supervisor call (SVC) interruption.

The format of an SVC trace record depends on the SVC interruption being traced. For a break down of the information that GTF collects for each SVC, see the *SVC Summary* chapter of *z/OS MVS Diagnosis: Reference*. The formats shown are typical.

Minimal Trace Record Format

```
SVC   CODE.... hhh  ASCB.... hhhhhhhh CPU..... hhhh     PSW..... hhhhhhhh
                             hhhhhhhh TCB..... hhhhhhhh R15..... hhhhhhhh
                    R0...... hhhhhhhh R1...... hhhhhhhh
```

Comprehensive Trace Record Format

```
SVC..... hhh   ASCB.... hhhhhhhh CPU..... hhhh     JOBNAME. cccccccc
               OLD-PSW. hhhhhhhh hhhhhhhh          TCB..... hhhhhhhh
               MODN.... yyyyyyyy
```

SVC CODE hhh
SVC hhh
SVC interruption code, which is also called the SVC number.

ASCB hhhhhhhh
Address of the ASCB for the address space in which the interruption occurred.

CPU hhhh
Address of the processor on which the interruption occurred.

JOBNAME cccccccc
One of the following:

cccccccc	Name of the job associated with SVC interruption
SSSSSSSS	Unavailable; GTF cannot provide data for the SVC due to security considerations
********	An internal error occurred

PSW hhhhhhhh hhhhhhhh
OLD-PSW hhhhhhhh hhhhhhhh
Program status word stored when the interruption occurred.

TCB hhhhhhhh
Address of the TCB for the interrupted task, that is, the task that issued the SVC instruction.

R15 hhhhhhhh
R0 hhhhhhhh
R1 hhhhhhhh
Data in general registers 15, 0, and 1 when the SVC instruction ran.

MODN cccccccc
cccccccc is one of the following:

mod_name
The name of a module that will receive control when the task is dispatched.

SVC–T2
Indicates a type 2 SVC routine resident in the nucleus.

SVC–RES
Indicates a type 3 SVC routine or the first load module of a type 4 SVC routine. The routine is located in the pageable link pack area (PLPA).

SVC cccc
Indicates the second or subsequent load module of a type 4 SVC routine. The routine is located in the fixed or pageable link pack area (LPA). The last four characters of the load module name are cccc.

****IRB******
Indicates an asynchronous routine with an associated interruption request block. No module name is available.

***cccccc**
Indicates an error recovery module. The last seven characters of the load module name are cccccc.

PPPPPPPP
A page fault occurred

****
An internal error occurred

DDNAM cccccc
Name of the DD statement associated with the SVC, if applicable.

additional fields
Vary with the SVC number. These fields are described for the SVC in the *z/OS MVS Diagnosis: Reference*.

USR Trace Records

Purpose

A USR record represents processing of a GTRACE macro. A user-supplied formatting routine (AMDUSRhh) formats the record. If a routine is not supplied, GTF prints the record without formatting.

This topic shows the unformatted and formatted records, then shows the following examples of USR records created by GTRACE macros in IBM-components:
- USRF9 trace records for VSAM
- USRFE trace records for BSAM, QSAM, BPAM, and BDAM
- USRFF trace records for open, close, and end-of-volume (EOV)

The USRFD trace records for VTAM are described in *z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures*.

USR records contain the following information useful for identifying the user program, MVS component, or IBM product producing the record and the routine you can use to format the record:
- Event identifiers (EIDs) identify the event that produced the record. See "Event Identifiers (EIDs) for USR Trace Records" on page 10-75 for a list of the EIDs and associated products for USR trace records. Because each EID for USR records start with an E, unformatted USR records show just the last three numbers of the EID after the E.
- Format identifiers (FIDs) identify the routine that the system used to format the USR trace record. See "Format Identifiers (FIDs) for USR Trace Records" on page 10-76 for a list of the FIDs and associated routines.

Unformatted USR Trace Record

Purpose

An unformatted user trace record represents processing of a GTRACE macro when a formatting routine is not supplied.

Record Format

```
USR AID hh FID hhhh EID hhhh hhhhhhhh hhhhhhhh ....
```

AID hh
Application identifier, which should always be AID FF.

FID hhhh
Format identifier of the routine (AAMDUSRhh) that was to format this record. See "Format Identifiers (FIDs) for USR Trace Records" on page 10-76 for a list of the FIDs and associated formatting routines for user trace records.

EID hhhh
Event identifier, which identifies the event that produced the record. See "Event Identifiers (EIDs) for USR Trace Records" on page 10-75 for a list of the EIDs and associated products for USR trace records.

hhhhhhhh hhhhhhhh ....
Recorded data (268 bytes maximum). The data are as follows:
- Bytes 0-3: ASCB address
- Bytes 4-11: jobname
- Bytes 12-256: user data

Formatted USR Trace Record

Purpose

A formatted user trace record represents processing of a GTRACE macro when an AMDUSRhh formatting routine is supplied.

Record Format

```
USRhh hhh ASCB hhhhhhhh JOBN cccccccc
          xxxx ...
```

USRhh
Identifies the user-supplied formatting routine (AMDUSRhh). The following USR records are generated and formatted by system components, and are described in the following topics:
- USRF9 Trace Records for VSAM
- USRFD Trace Records for VTAM
- USRFE Trace Records for BSAM, QSAM, BPAM, and BDAM
- USRFF Trace Records for Open/Close/EOV

hhh
Last three numbers of the event identifier (EID) specified in the GTRACE macro. See "Event Identifiers (EIDs) for USR Trace Records" on page 10-75 for a list of the EIDs and associated products for USR trace records.

ASCB hhhhhhhh
Address of the ASCB for the address space that created the record.

JOBN cccccc
Name of the job associated with the address space.

xxxx ...
User-formatted trace data.

USRF9 Trace Record for VSAM

Purpose

A USRF9 trace record represents opening or closing of a VSAM data set.

Record Format

```
USRF9 FF5   ASCB hhhhhhhh JOBN cccccc
            JOB NAME cccccc STEP NAME cccccc
            TIOT ENT hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

            ACB      hhhhhhhh hhhhhhhh hhhhhhhh...
                     hhhhhhhh hhhhhhhh hhhhhhhh...

            AMBL     hhhhhhhh hhhhhhhh hhhhhhhh...
                     hhhhhhhh hhhhhhhh hhhhhhhh...

            AMB      hhhhhhhh hhhhhhhh hhhhhhhh...
                     hhhhhhhh hhhhhhhh hhhhhhhh...

            AMDSB    hhhhhhhh hhhhhhhh hhhhhhhh...
                     hhhhhhhh hhhhhhhh hhhhhhhh...

            AMB      hhhhhhhh hhhhhhhh hhhhhhhh...
                     hhhhhhhh hhhhhhhh hhhhhhhh...

            AMDSB    hhhhhhhh hhhhhhhh hhhhhhhh...
                     hhhhhhhh hhhhhhhh hhhhhhhh...
```

USRF9
Identifies VSAM’s trace-record formatting routine (AMDUSRF9).

FF5
Last three numbers of the event identifier (EID) specified in the GTRACE macro. See “Event Identifiers (EIDs) for USR Trace Records” on page 10-75 for a list of the EIDs and associated products for USR trace records.

ASCB hhhhhhhh
Address of the ASCB for the address space in which the event occurred.

JOBN cccccc
JOB NAME cccccc
Name of the job.

STEP NAME cccccc
Name of the job step during which the event occurred.

TIOT ENT hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
Data set entry from the task I/O table (TIOT).

ACB hhhhhhhh ...
Contents of the data set’s access method control block (ACB).

AMBL hhhhhhhh ...
Contents of the AMB list (AMBL).

AMB hhhhhhhh ...
Contents of the access method block (AMB). The first AMB is for data, the second for the index.

AMDSB hhhhhhhh ...
Contents of the access method statistics block (AMDSB). The first AMDSB is for data, the second for the index.

USRFD Trace Record for VTAM

Reference

See *z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures* for samples of the USRFD trace records.

USRFE Trace Record for BSAM, QSAM, BPAM, and BDAM

Purpose

A USRFE trace record represents abnormal termination of an access method routine for BSAM, QSAM, BPAM, or BDAM.

Record Format

```
USRFE hhh ASCB hhhhhhhh JOBN cccccc

      BSAM/QSAM/BPAM/BDAM TRACE RECORD DDNAME cccccc ABEND CODE hh

      cccc...[AT LOCATION hhhhhhhh]
      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh ...
      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh ...
```

USRFE
Identifies the trace-record formatting routine (AMDUSRFE).

hhh
Last three numbers of the event identifier (EID) specified in the GTRACE macro. See “Event Identifiers (EIDs) for USR Trace Records” on page 10-75 for a list of the EIDs and associated products for USR trace records. The event identifier (EID) corresponds to the system completion code as follows:

EID	Code
FF3	002
FF4	008
FF6	112
FF7	215
FF8	119
FF9	235
FFA	239
FFB	145
FFC	251
FFD	451
FFE	169

ASCB hhhhhhhh
Address of the ASCB for the address space in which the abnormal termination occurred.

JOBN cccccc
Name of the job associated with the address space.

BSAM/QSAM/BPAM/DBAM TRACE RECORD
Record identification provided by the AMDUSRFE formatting routine.

DDNAME cccccc
Name of the DD statement for the data set being processed.

ABEND CODE hhh
System completion code for the abnormal termination of the task.

RETURN CODE hh
Return code from the module that detected the error condition.

TIME=dd.dd.dd
Time (hour.minute.second) when the GTRACE macro was processed or blank, if the time is not available.

ccc...[AT LOCATION hhhhhhhh]
hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh ...
Data area name, or name and address, followed by the data area contents.

USRFF Trace Record for Open/Close/EOV Abnormal End

Purpose

This USRFF trace record represents an abnormal end during open, close, or end-of-volume (EOV).

Record Format

```
USRFF   FFF    ASCB hhhhhhhh JOBN cccccc
               xxxx ...
```

USRFF
Identifies the Open/Close/EOV trace record formatting routine (IMDUSRFF).

FFF
Last three numbers of the event identifier (EID) specified in the GTRACE macro. See "Event Identifiers (EIDs) for USR Trace Records" on page 10-75 for a list of the EIDs and associated products for USR trace records.

xxxx ...
Unformatted RRCBSA's (recovery routine control block save areas).

USRFF Trace Record for User-Requested Work Area

Purpose

This USRFF trace record represents a user request for a work area trace.

Record Format

```
USRFF     FFF     ASCB hhhhhhhh JOBN cccccc
DCB       xxxx ...
WKAREA1   xxxx ...
WKAREA2   xxxx ...
WKAREA3   xxxx ...
WKAREA4   xxxx ...
WKAREA5   xxxx ...
WTG TBL   xxxx ...
```

USRFF
Identifies the Open/Close/EOV trace record formatting routine (IMDUSRFF).

FFF
Last three numbers of the event identifier (EID) specified in the GTRACE macro. See “Event Identifiers (EIDs) for USR Trace Records” for a list of the EIDs and associated products for user trace records.

DCB
Data control block.

WKAREA1
Volume labels, file labels, DSCBs or message area. See *z/OS DFSMS: Using Magnetic Tapes*.

WKAREA2
Job file control block.

WKAREA3
Internal control blocks for Open/Close/EOV. These blocks are the data control block (DCB), data extent block (DEB), and the input/output block (IOB).

WKAREA4
WKAREA5
Where-to-go-table used in transferring control among CSECTs of Open/Close/EOV.

Event Identifiers (EIDs) for USR Trace Records

The event identifier (EID) in GTF trace records is a 2-byte hexadecimal number that identifies the event producing the record. You can use it to identify the product that produced the record.

This table shows the full 2-byte EID, but because EIDs for USR records start with an E, often unformatted USR records show just the last three numbers of the EID after the E. If you have a three number EID, such as FF5, look for EFF5 in the table below.

EID (hex)	Symbolic Name	Issued by
E000-E3FF		GTF user program
E400-E5F0		Reserved for IBM use
E5F1		PVM
E5F2-E5F3		Reserved for IBM use
E5F4-E5F5		NetView® System Monitor
E5F6-EF43		Reserved for IBM use
E544-EF45		RACF
EF46-EF47		Reserved for IBM use
EF48		IOS
EF49		BDT
EF4F		OSAM
EF50-EF52		Reserved for IBM use
EF53		OSI
EF54-EF5D		FSI
EF5E		Reserved for IBM use
EF5F		DB2
EF60		JES3
EF61		VSAM Buffer Manager
EF62		Dynamic output SVC installation exit

EID (hex)	Symbolic Name	Issued by
EF63		Converter/Interpreter installation exit
EF64		APPC/VM VTAM Support (AVS)
EF65		GETMAIN FREEMAIN STORAGE trace (MVS)
EF66-EF6A		VTAM
EF6C		CICS
EFA0-EFA9		TCAM
EFAA		VTAM VM/SNA Console Services (VSCS)
EFAB-EFAE		Reserved for IBM use
EFAF-EFE0	IMDGPD01-IMDGPD50	IBM
EFE1	ISTVIEID	VTAM
EFE2	ISTTHEID	VTAM
EFE3	ISTTREID	VTAM
EFE4	ISTTDEID	VTAM
EFE5-EFEE		JES2
EFEF	ISTTPEID	VTAM
EFEF	ISTTPEID	VTAM
EFF0	ISTRPEID	VTAM
EFF1	ISTCLEID	VTAM
EFF2	ISTLNEID	VTAM
EFF3	IGGSP002	SAM/PAM/DAM
EFF4	IGGSP008	SAM/PAM/DAM
EFF5	IDAAM01	VSAM
EFF6	IGGSP112	SAM/PAM/DAM
EFF7	IGGSP215	SAM/PAM/DAM
EFF8	IGGSP119	SAM/PAM/DAM
EFF9	IGGSP235	SAM/PAM/DAM
EFFA	IGGSP239	SAM/PAM/DAM
EFFB	IGGSP145	SAM/PAM/DAM
EFFC	IGGSP251	SAM/PAM/DAM
EFFC	IGGSP451	SAM/PAM/DAM
EFFE	IGGSP169	SAM/PAM/DAM
EFFF	IHLMDMA1	OPEN/CLOSE/EOV

Format Identifiers (FIDs) for USR Trace Records

The format identifier (FID) in GTF trace records is a one-byte hexadecimal number that is used to determine the name of the GTFTRACE module you can use to format USR records. See *z/OS MVS IPCS Customization* for information about the GTFTRACE formatting appendage for formatting USR trace records.

FID (hex)	EID	Issued by	Optional format module
00	E000-EFE4	User/component	CSECT AHLFFILT in AHLFINIT
01-50	E000-E3FF	User	IMDUSR or AMDUSR (01-50)
57	EF44-EF45	RACF	AMDUSR57
81		VMSI	IMDUSR81 or AMDUSR81
84		VMSI/VTAM	IMDUSR84 or AMDUSR84
DC		PVM	IMDUSRDC or AMDUSRDC
E2-E3		PSF/MVS	IMDUSRE2-IMDUSER3 or AMDUSRE2-AMDUSER3
E6		OSI	IMDUSRE6 or AMDUSRE6
E8		FSI	IMDUSRE8 or AMDUSRE8

FID (hex)	EID	Issued by	Optional format module
E9		DB2/VSAM	IMDUSRE9 or AMDUSRE9
EB		APPC/VM VTAM Support (AVS)	IMDUSREB or AMDUSREB
EC		VTAM	IMDUSREC or AMDUSREC
F5		VTAM/VSCS	IMDUSRF5 or AMDUSR5
F7		ACF/TCAM	IMDUSRF7 or AMDUSRF7
F9	EFF5	VSAM	IMDUSRF9 or AMDUSRF9
FD	EFEF-EFF2	VTAM	IMDUSRFD or AMDUSRFD
FE	EFF3-EFF4 EFF6-EFFE	SAM/PAM/DAM	IMDUSRFE or AMDUSRFE
FF	EFFF	OPEN/CLOSE/EOV	IMDUSRFF or AMDUSRFF

Unformatted GTF Trace Output

This topic describes GTF output records that are not formatted by IPCS or other routines. You can use this information to write your own formatting or analysis routines.

Note: When GTF cannot obtain the data normally placed in fields of the following records, it signals this by placing one of the following values in the field
- C'U/A'. Blanks are added on the right to fill out the field.
- C'*'. Asterisks are replicated to fill out the field.

There are several types of output records:
- Control records, see "Control Records".
- Lost data records, see "Unformatted Lost Event Records" on page 10-79.
- User data record, see "User Data Records" on page 10-80.
- System data records, see "System Data Records" on page 10-81.

The lost data, user data and system data records all contain optional fields, which are fields that only appear under certain conditions. The conditions are covered in the explanation for the fields. Make sure that your formatting or analysis routine takes these variable fields into account.

This section also describes the GTF system data records for individual events. See "CCW Trace Record" on page 10-82 through "SVC Minimal Trace Record" on page 10-96.

Control Records

GTF creates a control record at the start of each block of trace output. The control record can be followed by lost data, user data, and system data records. If this trace output was merged from multiple systems using the IPCS COPYTRC subcommand, then the control record reflects the combined GTF options in effect from all the systems.

Reference

See *z/OS MVS IPCS Commands* for more information about the COPYTRC subcommand.

Figure 10-4 shows the format of a control record.

```
|length| res | AID | FID |time zone|time stamp|options| ... source descriptors ...
   2      2     1     1       4          8         8
 bytes  bytes  byte  byte   bytes      bytes     bytes
```

Figure 10-4. Unformatted Control Record

The fields in the control record contain the following information:

length
Total length of the record, in bytes.

res
Two bytes of zeroes. Reserved for IBM use.

AID
Application identifier, which is always zero for control records.

FID
Format identifier of the routine that will format the record, which is always X'01' for a control record.

time zone
Value showing the difference between local time and Greenwich mean time (GMT) in binary units of 1.048576 seconds.

time stamp
Time stamp showing the eight-byte Greenwich mean time (GMT) when the control record was created.

options
An eight-byte field containing the following:

The first five bytes identify the GTF options in effect for a block of trace output. See mapping macro AHLZGTO in *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)*.

The remaining 3 bytes contain the following important flags, in bit ranges 0-7:

GTWCFSID - Byte 6, Bit 6
1, if the individual trace records have SIDs (system identifiers) indicating that the GTF trace data from multiple systems was merged using the IPCS COPYTRC command. In this case, there will be multiple source descriptors, one for each system. The source descriptors are ranged in order by system identifier (SID). Use the value in the SID field as an array index to locate the source descriptor for a particular system.

The source descriptor information will be identical in all control records within a single trace data set.

0, if the trace records have no SIDs.

GTWCFNEW - Byte 6, Bit 7
1, if this block of trace data was written by a MVS/ESA SP™ Version 4 or later system.

0, if this block of data was written by a pre-MVS/ESA Version 4 system.

Source descriptors
One or more arrays of information about the origins of the records in this block of trace data, such as the release level of the system issuing the trace data and

the GTF options in effect. If GTF trace data was merged from multiple systems, there will be multiple source descriptors, one for each system. Use the value in the SID field as an array index to locate the source descriptor for a particular system.

See mapping macro AHLZGTS in *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)* to see the format of the source descriptor information.

Unformatted Lost Event Records

A lost event record indicates that GTF lost the trace records for one or more events because of an error or overflow of the trace buffer. Figure 10-5 shows the format of a lost event record.

length	res	AID	FID	time zone	time stamp	count	SID
2 bytes	2 bytes	1 byte	1 byte	4 bytes	8 bytes	4 bytes	2 bytes (optional)

Figure 10-5. Unformatted Lost Event Record

The fields in the lost event record contain the following information:

length
Total length of the record, in bytes.

res
Two bytes of zeroes. Reserved for IBM use.

AID
Application identifier, which is always zero for lost event records.

FID
Format identifier. The value of FID is one of the following:
- X'02', if some trace records are missing because of an error or an overflow of the trace buffer.
- X'03', if an entire block of trace records is missing because of an error or an overflow of the trace buffer.

time zone
Value showing the difference between local time and Greenwich mean time (GMT) in binary units of 1.048576 seconds.

time stamp
Time stamp showing the eight-byte Greenwich mean time (GMT) when the control record was created.

count
If the FID is X'02', indicating that some trace records are missing, this field contains the number of trace events that are lost.

If the FID is X'03', indicating that an entire block of trace data is missing, this field contains zeros.

SID
The system identifier of the system where this trace record was created.

This 2-byte field only exists when GTF trace data from multiple systems was merged using the IPCS COPYTRC command. When present, the SID is an array index you can use to locate the source descriptor information for a particular system. For example, if the SID value for a record is 3, the source descriptor information for the system issuing the record will be the third source descriptor in the control record.

To check to see whether trace data for a block of output comes from multiple systems, look in the control record for the **options** field and see if the GTWCFSID bit is set on. See 10-78 for the **options** field.

User Data Records

This topic describes the format of user trace records requested using the GTRACE macro.

If the application using GTRACE specifies more than 256 bytes of data, the user records may be split. If a user trace record is a split record, the AID will contain a value of X'F0', X'F1', or X'F2'. Split records contain the optional **sequence** and **total length** fields.

The records have the general format shown in Figure 10-6.

length	res	AID	FID	time stamp	EID	SID	sequence	total length	ASCB	job name	... data ...
2 bytes	2 bytes	1 byte	1 byte	8 bytes	2 bytes	2 bytes (optional)	2 bytes (optional)	4 bytes (optional)	4 bytes	8 bytes	

Figure 10-6. Unformatted User Trace Record Format

The fields in the record contain the following information:

length
Total length of the record, in bytes.

res
Two bytes of zeros. Reserved for IBM use.

AID
Application identifier, which is one of the following:
- X'FF'-- Non-split record
- X'F0'-- The first record of a series of split records
- X'F1'-- A middle record in a series of split records
- X'F3'-- The last record in a series of split records

FID
Format identifier of the routine that will format the trace record. See “Format Identifiers (FIDs) for USR Trace Records” on page 10-76 for a list of FIDs and associated formatting routines.

time stamp
Time stamp showing the eight-byte Greenwich mean time (GMT) when the record was created.

EID
Event identifier, which identifies the event that produced the trace record. See “Event Identifiers (EIDs) for USR Trace Records” on page 10-75 for a list of the EIDs and associated products for user trace records.

SID
System identifier, which identifies the system where the record was produced. This 2-byte field only exists in the following cases:

- GTF trace data from multiple systems was merged using the IPCS COPYTRC command.
- The record is a split one. If the trace data containing this split record was not merged from multiple systems, the SID field for the split record contains zeros.

You can use the SID from merged trace data as an array index to locate the source descriptor information for a particular system. For example, if the SID value for a record is 3, the source descriptor information for the system issuing the record will be the third source descriptor in the control record.

To check to see whether trace data for a block of output comes from multiple systems, look in the control record for the **options** field and see if the GTWCFSID bit is set on. See 10-78 for the **options** field.

sequence
Sequence number, in hexadecimal, of this split record. This field only exists for split records.

total length
Total length of the split trace data. This field only exists for split records.

ASCB
The address of the address space control block (ASCB) for the address space where the GTRACE macro was issued.

jobname
The name of the job associated with the task where the GTRACE macro was issued.

data
Contains the trace data gathered for the requested event. The length of this field varies according to the event being traced. The number of bytes of data in the data field for user records is equal to the number of bytes specified on the GTRACE macro.

System Data Records

GTF creates trace records for each system event you select when requesting GTF tracing. The header portion of system data records for events is shown in Figure 10-7. Individual event record formats The format of individual system data records are shown in “Unformatted Trace Records for Events” on page 10-82 in alphabetical order. Note that this section does not include all system events.

length	res	AID	FID	time stamp	EID	SID	... data ...
2 bytes	2 bytes	1 byte	1 byte	8 bytes	2 bytes	2 bytes (optional)	

Figure 10-7. Header for Unformatted System Trace Record Format

The fields in the record contain the following information:

length
Total length of the record, in bytes.

res
Two bytes of zeros. Reserved for IBM use.

AID
Application identifier, which is always X'FF' for system data records.

FID
Format identifier of the routine that will format the trace record.

time stamp
Time stamp showing the eight-byte Greenwich mean time (GMT) when the record was created.

EID
Event identifier, which identifies the event that produced the trace record.

SID
System identifier, which identifies the system where the record was produced. The SID field contains zeros when the record is a split record. This 2-byte field is only created when GTF trace data from multiple systems was merged using the IPCS COPYTRC command. When present, the SID is an array index you can use to locate the source descriptor information for a particular system. For example, if the SID value for a record is 3, the source descriptor information for the system issuing the record will be the third source descriptor in the control record.

To check to see whether trace data for a block of output comes from multiple systems, look in the control record for the **options** field and see if the GTWCFSID bit is set on. See 10-78 for the **options** field.

data
Trace data gathered for the requested event. The length of this field varies according to the event being traced.

The data portions for individual system trace records are shown starting on "Unformatted Trace Records for Events".

Unformatted Trace Records for Events

This topic presents the records for a selection of system events in alphabetical order. It shows the unformatted layout of the data for individual event records. See "System Data Records" on page 10-81 to see the header section for the records. Note that not all system events are included in this topic.

Fields in a trace record may contain the following special indicators:

N/A	Not applicable. The field does not apply in this record. In a 2-byte field, not applicable appears as `N/`.
U/A	Unavailable. GTF could not gather the information. In a 2-byte field, unavailable appears as `U/`.

The offsets for all the data records are relative and do not reflect the actual number of bytes into the record for each field. The offsets begin at the start of the data portion of the each record because the header section varies in length, depending on whether the optional SID field is present.

CCW Trace Record

GTF builds a CCW record when a start subchannel (SSCH) occurs in the I/O supervisor SSCH subroutine or an I/O interruption occurs and TRACE=CCW or TRACE=CCWP GTF options are in effect. To trace channel programs, TRACE=SSCH or TRACE=I/O must also be in effect.

The FID for the CCW trace record is X'07'.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname.
14 (E)	2	Device address.
16 (10)	2	Maximum amount of data for each CCW.
18 (12)	1	Reserved for IBM use.
19 (13)	2	Sequential record count for this event.
21 (15)	1	Reserved for IBM use.
22 (16)	2	Flags describing data.
24 (18)	2	Length of data in this CCW.
26 (1A)	1	Length of data in this record.
27 (1B)	4	Virtual address of CCW.
31 (1F)	8	CCW being traced.
39 (27)	217	Data area or repeat of offset 22-39 if space available.

DSP Comprehensive Trace Record

GTF builds a DSP record when an entry is made to the dispatcher to dispatch a unit of work and TRACE=DSP is the GTF option in effect.

The FID for the DSP comprehensive trace record is X'00'. The EID is one of the following:

- X'0001' - indicates SRB dispatching.
- X'0002' - indicates LSR dispatching.
- X'0003' - indicates TCB dispatching.
- X'0004' - indicates exit prolog dispatching.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname.
14 (E)	8	Resume PSW for new task.
22 (16)	4	New TCB (for LSR and TCB, SRB for SRB).
For TCB only:		
26 (1A)	8	CDE name.
For SRB only:		
26 (1A)	4	Parm address.
30 (1E)	1	For SRB only, SRB routine type indicator. S for a suspended SRB that is about to be re-dispatched. I for an SRB that is about to be dispatched for the first (initial) time.

DSP Minimal Trace Record

GTF builds a DSP minimal record when an entry is made to the dispatcher to dispatch a unit of work and both TRACE=SYSM,DSP are the GTF options in effect.

The FID for the DSP minimal trace record is X'03'. The EID is one of the following:

- X'0001' - indicates SRB dispatching.
- X'0002' - indicates LSR dispatching.
- X'0003' - indicates TCB dispatching.
- X'0004' - indicates exit prolog dispatching.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Resume PSW for work unit being dispatched.
14 (E)	4	Current TCB or N/A (for TCB and LSR only).
18 (12)	4	Register 15.
22 (16)	4	Register 0 or SRB.
26 (1A)	4	Register 1.
30 (1E)	1	For SRB only, SRB routine type indicator. S for a suspended SRB that is about to be re-dispatched. I for an SRB that is about to be dispatched for the first (initial) time.

EXT Comprehensive Trace Record

GTF builds a EXT comprehensive record when an external interruption occurs and either TRACE=SYS or TRACE=EXT are the GTF options in effect.

The FID for the EXT comprehensive trace record is X'02'. The EID is X'6201'. interruption.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname.
14 (E)	8	External old PSW.
22 (16)	4	Old TCB.
For SIGP interrupt:		
26 (1A)	4	PARMFIELD.
30 (1E)	2	CPUID.
for clock comparator interrupt:		
26 (1A)	2	Reserved for IBM use.
28 (1C)	4	TQE exit.
32 (20)	4	TQE ASCB.
For CPU timer interrupt:		
26 (1A)	2	Reserved for IBM use.
28 (1C)	4	TQE exit.

EXT Minimal Trace Record

GTF builds an EXT minimal record when an external interruption occurs and TRACE=SYSM is the GTF option in effect.

The FID for the EXT minimal trace record is X'03'. An EID of X'6201' indicates an external interruption.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	External old PSW.
14 (E)	4	TCB of interrupted task or N/A.
18 (12)	4	NTQE TCB or INT CPU or N/A.

I/O Summary Trace Record

GTF builds an I/O summary record when an I/O interruption occurs and TRACE=IOX or TRACE=IOXP is a GTF option in effect. To trace PCI I/O interruptions, TRACE=PCI must also be in effect.

The FID for the I/O summary record is X'08'. The EID is one of the following:

- X'2100' - indicates a PCI I/O interruption
- X'5107' - indicates an EOS I/O interruption
- X'5202' - indicates an I/O interruption with a valid UCB
- X'5203' - indicates a CS I/O interruption. It indicates an end of sense interruption for a device containing the concurrent sense facility.

The I/O summary record will always contain a header section, followed by a section header and a common section. The section header describes the type and length of the following section and an indicator if this is the last section of the record.

A typical I/O summary record for a dasd device would have a header section, a common section, a data set section, a CMB section and, probably, one or more CCW sections. If an I/O summary record has to be extended the following extension records would consist of a header section with the header record number greater than 1, a common section and one or more CCW sections.

Header Section:

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname.
14 (E)	2	Device number.
16 (10)	4	System ID.
20 (14)	1	Driver ID.
21 (15)	1	Trace version.
22 (16)	2	Record count.

Section Header:

Offset	Size	Description
0 (0)	1	Section type: FL1'0': Common section FL1'1': CMB section FL1'3': CCW Orientation section FL1'4': Data set section
1 (1)	1	Flag identifiers.
	1...	Last section of this record.

Offset	Size	Description
2 (2)	2	Section length.

Common Section:

Offset	Size	Description
0 (0)	1	Device class.
1 (1)	1	Device status.
2 (2)	1	Error codes: indicate errors found during CCW analysis.
3 (3)	1	Flag byte.
	1...	Last trace recordof this I/O event.
	.1..	Reserve (conditional or unconditional).
	..1.	Release.
	...1	At least one search CCW.
4 (4)	6	Volume serial.
10 (A)	4	Device type.

Data Set Section:

Offset	Size	Description
0 (0)	1	Data set type.
1 (1)	1	Name length.
2 (2)	44	Data set name.

CMB Section:

Offset	Size	Description
0 (0)	2	Number of SSCH instructions.
2 (2)	2	Number of SSCH instructions for which data was collected.
4 (4)	4	Sum of device connect times.
8 (4)	4	Sum of SSCH request pending times.
12 (C)	4	Sum of subchannel disconnect times.
16 (10)	4	Sum of control unit queueing times.
20 (14)	8	Time stamp for the start of this I/O request.
28 (1C)	4	Device active only time.

CCW Orientation Section:

Offset	Size	Description
0 (0)	1	Orientation sequence number.
1 (1)	1	Flag byte 1.
	1...	At least one SILI bit on.
	.1..	At least one suspend bit on.
	..1.	At least one PCI.
	...1	At least one skip bit on.
	.... 1...	Read record zero or read home address.
	.... .1..	Reserved.
	.... ..1.	Read multiple CKD or track.
	1	At least one erase CCW.
2 (2)	1	Flag byte 2.
	1...	End of file read or written.
	.1..	At least one format write.

Offset	Size	Description
3 (3)	1	Count of erase, read MCKD.
4 (4)	2	Number of blocks read.
6 (6)	2	Number of blocks written.
8 (8)	4	Number of bytes read.
12 (C)	4	Number of bytes written.
16 (10)	2	Number of data chain CCWs.
18 (12)	2	Number of COM chain CCWs.
20 (14)	1	External global attribute.
	11..	’11’ (only allowed value).
	..1.	CKD conversion mode.
	...1	Subsystem operation mode.
	.... ..1.	Cache fast write.
	1	Inhibit DASD fast write.
21 (15)	1	External global attribute extended.
22 (16)	4	External end of extent CCH.
26 (1A)	1	Seek/locate code.
27 (1B)	5	CCHHR (search ID equal).
32 (20)	1	Operation code from locate parameter.
33 (21)	1	Sector number from parameter.
34 (22)	1	Extended operation code.
35 (23)	2	Extended parameters.

I/O Trace Record

GTF builds an I/O record when an I/O interruption occurs and TRACE=SYSM, TRACE=SYS, TRACE=IO, or TRACE=IOP are the GTF options in effect. To trace PCI I/O interruptions, TRACE=PCI must also be in effect.

The FID for the I/O trace record is X'07'. The EID is one of the following:

- X'2100' - indicates a PCI I/O interruption.
- X'5101' - indicates an EOS I/O interruption.
- X'5200' - indicates an I/O interruption with a valid UCB.
- X'5201' - indicates a CS I/O interruption. It indicates an end of sense interruption for a device containing the concurrent sense facility.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname.
14 (E)	2	Device number.
16 (10)	8	I/O old PSW.
24 (18)	20	IRB words 1-5.
44 (2C)	4	TCB.
48 (30)	2	Sense bytes.
50 (32)	1	IOSB Flag (IOSFLA).
51 (33)	1	IOSB Option (IOSOPT).
52 (34)	1	IOS Driver ID.
53 (35)	1	IOS level.
54 (36)	1	UCB level.
55 (37)	1	Flags (UCBGWT).
56 (38)	2	Base device number.
58 (3A)	44	IRB words 6–16 (for EID X’5201’ only).

PI Comprehensive Trace Record

GTF builds a PI comprehensive record when a program interruption occurs and either TRACE=PI or TRACE=SYS are the GTF options in effect.

The FID for the PI comprehensive trace record is X'00'. The EID is one of the following:

- X'6101' - indicates a program interruption with codes 1-17, 19, and 128.
- X'6200' - indicates a program interruption with code 18.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname.
14 (E)	8	Program old PSW.
22 (16)	4	INT TCB.
26 (1A)	4	Virtual page address.
30 (1E)	8	RB or CDE name.
38 (26)	64	Reserved for IBM use.
102 (66)	4	Virtual page address high half.

PI Minimal Trace Record

GTF builds a PI minimal record when a program interruption occurs and TRACE=SYSM is the GTF option in effect.

The FID for the PI minimal trace record is X'03'. The EID is one of the following:

- X'6101' - indicates a program interruption with codes 1-17, 19, and 128.
- X'6200' - indicates a program interruption with code 18.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Program old PSW.
14 (E)	4	Old TCB.
18 (12)	4	Virtual page address.
22 (16)	4	Register 15.
26 (1A)	4	Register 1.
30 (1E)	4	Virtual page address high half.

RR Comprehensive Trace Record

GTF builds an RR comprehensive record when a recovery routine is invoked and TRACE=SYS or TRACE=RR are the GTF options in effect.

The FID for the RR comprehensive trace record is X'04'. The EID is one of the following:

- X'4002' - indicates STAE/ESTAE invocation.
- X'4003' - indicates FRR invocation.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname.
14 (E)	8	Name of recovery routine or U/A.
22 (16)	8	PSW current when error occurred.
30 (1E)	4	Completion code.

Offset	Size	Description
34 (22)	8	Reserved for IBM use.
42 (2A)	4	Retry address or N/A.
46 (2E)	4	Address of SDWA (STAE/ESTAE only).

RR Minimal Trace Record

GTF builds an RR minimal record when a recovery routine is invoked and TRACE=SYS is the GTF option in effect.

The FID for the RR minimal trace record is X'03'. The EID is one of the following:

- X'4002' - indicates STAE/ESTAE invocation.
- X'4003' - indicates FRR invocation.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Error PSW.
14 (E)	4	Completion code.
18 (12)	4	Reserved for IBM use.
22 (16)	3	Reserved for IBM use.
25 (19)	1	Return code from recovery routine (STAE/ESTAE only).
26 (1A)	4	Retry address or N/A. **Note:** If no return code at offset 41, begin return address at offset 41.
30 (1E)	4	Address of SDWA (STAE/ESTAE only). **Note:** If return address begins at offset 41, SDWA address begins at offset 45.

SLIP Trace Records

GTF builds a SLIP trace record when TRACE=SLIP is the GTF option in effect and:

- A SLIP trap has matched and either TRACE or TRDUMP has been specified on the SLIP command.
- A SLIP trap is in DEBUG mode (specified on the SLIP command) and is inspected by the SLIP processor as a result of any SLIP event.

The SLIP trace records are:

- SLIP Standard Trace Record
- SLIP Standard/User Trace Record
- SLIP User Trace Record
- SLIP DEBUG Trace Record

SLIP Standard Trace Record: The FID for the SLIP standard trace record is X'04'. The EID is X'4004'.

A field will contain asterisks if an error occurred when attempting to obtain data or the data is unavailable because it is paged out.

Offset	Size	Description
0 (0)	4	ASCB address.
4 (4)	2	CPUID. (**Note:** When SLIP is entered from RTM2, the CPUID recorded may be different from the CPUID when RTM2 was running.)
6 (6)	8	Jobname from current address space (or N/A).
14 (E)	4	SLIP trap ID.

Offset	Size	Description
18 (12)	2	ASID of current address space.
20 (14)	8	Job step program name (or U/A or N/A).
28 (1C)	4	TCB address (or N/A).
32 (20)	1	System mode indicators, byte 1:
	1...	Supervisor control mode.
	.1..	Disabled for I/O and external interrupts.
	..1.	Global spin lock held.
	...1	Global suspend lock held.
	.... 1...	Local lock held.
	1	Type 1 SVC in control.
	.... ..1.	SRB mode.
	1	TCB mode.
33 (21)	1	System mode indicators, byte 2:
	1...	Recovery routine in control (always zero if a PER interrupt).
	.1..	Problem program state.
	..1.	Supervisor state.
	...1	System key.
	.... 1...	Problem program key.
	.... .1..	Any global lock held.
	.... ..1.	Any lock held.
36 (24)	1	Error byte 1 (or zeros if a PER interrupt):
	1...	Program check interrupt.
	.1..	Restart interrupt.
	..1.	SVC error.
	...1	Abend; task issued SVC 13.
	.... 1...	Paging I/O error.
	.... .1..	Dynamic address translation error.
	.... ..1.	Software error caused by machine check.
	1	Abnormal address space termination.
35 (23)	1	Error byte 2 (or zeros if a PER interrupt):
	1...	Memterm.
36 (24)	1	SLIP flags:
	1...	DEBUG record.
	.1..	Registers collected.
37 (25)	2	Data unavailable counter (or zeros if DATA was not specified for the trap).

The following fields apply to PER interrupts only (otherwise set to N/A (or N or one-byte fields)).

Offset	Size	Description
39 (27)	8	Load module name in which the interrupt occurred (or U/A or N/A).
47 (2F)	4	Offset in load module (or U/A or N/A).
55 (37)	6	Instruction content (six bytes of data beginning at the address of the instruction that caused the PER interrupt).
61 (3D)	4	Target instruction address if EXECUTE instruction (or N/A or U/A).
65 (41)	66	Target instruction content if EXECUTE instruction (six bytes of data beginning at the target instruction address), or (N/A or U/A).

Offset	Size	Description
71 (47)	4	Beginning range virtual address if SA (storage-alteration) specified on SLIP command (or N/A).
75 (4B)	4	Four bytes of storage starting at beginning range virtual address if SA specified (or N/A or U/A).
79 (4F)	8	Program old PSW.
87 (57)	4	Program interrupt code (PIC) and instruction length code.
91 (5B)	1	PER interrupt code:
	1..	Successful-branch event (SB).
	.1..	Instruction-fetch event (IF).
	..1.	Storage-alteration event (SA).
92 (5C)	1	PER trap mode:
	1...	Successful-branch monitoring (SB).
	.1..	Instruction-fetch monitoring (IF).
	..1.	Storage-alteration monitoring (SA).
	...x	Reserved.
	.... 1...	PER trap.
	.... .1..	Recovery specified.
	.... ..1.	Message flag.
	1	Message flag.
93 (5D)	2	Keymask.
95 (5F)	2	SASID.
97 (61)	2	Authorization index.
99 (63)	2	PASID.
101 (65)	1	PSW ASC mode indicator F0: primary addressing mode F1: access register addressing mode F2: secondary addressing mode F3 home addressing mode
102 (66)	13	Storage Alteration Space Identifier For an address space: contains the ASID For a data space: contains the owning ASID and the dataspace name
115 (73)	4	PER interrupt code
	1	Reserved.
119 (77)	1	Reserved.

SLIP Standard + User Trace Record: The FID for the SLIP Standard + User trace record is X'04'. The EID is X'4005'.

Offset	Size	Description
0 (0)	4	
through 3		Fields are identical to the SLIP standard record.
115 (73)	5	Reserved.
120 (78)	2	Length of user-defined data.
122 (7A)	variable	User-defined data (specified via the TRDATA parameter on the SLIP command).

SLIP User Trace Record: The FID for the SLIP user trace record is X'04'. The EID is X'4006'.

Offset	Size	Description
0 (0)	2	CPUID.

Offset	Size	Description
2 (2)	2	Extension number.
4 (4)	1	Continuation length.
5 (5)	2	Length of the user defined data.
7 (7)	variable	User-defined data (specified via the TRDATA parameter on the SLIP command).

Notes:

1. If the SLIP user requests registers to be placed in the SLIP user record, they will be the first field in the record.
2. A length field of zero indicates that the user-defined data was not available (for example, the data is paged out).
3. The TRDATA parameter on the SLIP command specifies one or more data ranges. The number of records needed to trace these ranges depends on the size of the ranges specified. The trace contains a standard plus (+) user record from the next range or a user record followed by as many user records and user continuation records as needed to trace the ranges specified. The header for each record contains the CPUID and extension number to help correlate the output (extension numbers apply only to user and user continuation records). When a record is partially filled and the data from the next range will not fit in the remaining space; the partially filled record is written to the trace data set. Another user record is built, the extension number is increased by one, and the continuation length is set to zero. The data length and data is then copied into this record.

   When the length of the data from a range is greater than 249 bytes, the excess data is put in user continuation records in the following manner. The data length and first 248 bytes are put in a user record. After writing that record a user continuation record is built. The extension number is increased by one and the continuation length is set to the number of bytes of data to be put in this record. If more than 251 bytes of data are left, 248 bytes are copied into record, and it is written. User continuation records are built until all the data in from that range is traced.

SLIP DEBUG Trace Record: The FID for the SLIP DEBUG trace record is X'04'. The EID is X'4005'.

Offset	Size	Description
0 (0)	4	
through 3		Fields are identical to the SLIP standard record.
93 (5D)	27	

Offset	Size	Description
120 (78)	1	DEBUG byte; indication of which keyword failed: Decimal 2 indicates COMP keyword Decimal 3 indicates ASID keyword Decimal 4 indicates JOBNAME keyword Decimal 5 indicates JSPGM keyword Decimal 6 indicates PVTMOD keyword Decimal 7 indicates LPAMOD keyword Decimal 8 indicates ADDRESS keyword Decimal 9 indicates MODE keyword Decimal 10 indicates ERRTYP keyword Decimal 13 indicates RANGE keyword Decimal 14 indicates DATA keyword Decimal 20 indicates ASIDSA keyword Decimal 22 indicates REASON CODE keyword Decimal 23 indicates NUCMOD keyword Decimal 24 indicates PSWASC keyword Decimal 26 indicates DSSA keyword
121 (79)	1	Reserved.

Note: The high-order bit in the SLIP flags (SFLG) field (at offset X'34') is set on to indicate a DEBUG record.

SRM Comprehensive Trace Record

GTF builds an SRM comprehensive record when system resource manager is invoked and TRACE=SYS or TRACE=SRM are the trace options in effect.

The FID for the SRM trace record is X'04'. The EID is X'4001'.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname
14 (E)	4	Register 15.
18 (12)	4	Register 0.
22 (16)	4	Register 1.

SRM Minimal Trace Record

GTF builds an SRM minimal record when system resource manager is invoked and TRACE=SYSM is the GTF option in effect.

The FID for the SRM minimal trace record is X'03'. The EID is X'4001'.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	4	Register 15.
10 (A)	4	Register 0.
14 (E)	4	Register 1.

SSCH Trace Record

GTF builds an SSCH record when an SSCH event occurs and TRACE=SYSM, TRACE=SYS, TRACE=SYSP, TRACE=SSCH or TRACE=SSCHP are the GTF options in effect.

The FID for the SSCH trace record is X'00'. The EID is X'5105'.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname.
14 (E)	2	Device number.
16 (10)	4	Real address of channel program.
20 (14)	4	Virtual address of channel program.
24 (18)	4	Reserved for IBM use.
28 (1C)	1	Condition code.
29 (1D)	12	Reserved for IBM use.
41 (29)	8	Dynamic seek address.
49 (31)	1	Reserved for IBM use.
50 (32)	1	Reserved for IBM use.
51 (33)	1	Reserved for IBM use.
52 (34)	1	Reserved for IBM use.
53 (35)	1	Reserved for IBM use.
54 (36)	1	Reserved for IBM use.

SVC Comprehensive Trace Records

GTF builds SVC comprehensive records when an SVC interruption occurs and either the TRACE=SYS or TRACE=SVC GTF option is in effect. All SVC records contain the basic data described below; however, many SVC numbers invoke additional data recording, described following the basic data.

The FID for the SVC comprehensive trace record is X'010'. The EID is X'1000'.

Basic SVC Comprehensive Trace Record

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	Jobname.
14 (E)	8	SVC old PSW. The third and fourth bytes contain the SVC number.
22 (16)	4	Old TCB.
26 (1A)	8	CDE name.
34 (22)	4	Register 15.
38 (26)	4	Register 0.
42 (2A)	4	Register 1.

GTF builds only a basic comprehensive trace record for the following SVCs:

Name	Number	Name	Number
EXIT	3	TEST	97
GETMAIN/FREEMAIN	10	SUBMIT	100
TIME	11	QTIP	101
SYNCH	12	XLATE	103
MGCR	34	TOPCTL	104
WTL	36	IMBLIB	105
TTROUTER	38	REQUEST	106
CIRB	43	MODESET	107
TTIMER	46	None	109
TTOPEN	49	DSTATUS	110

Name	Number	Name	Number
NOP	50	JECS	111
OLTEP	59	RELEASE	112
TSAV	61	SIR	113
CHATR	72	BLKPAGE	115
(IFBSTAT)	76	None	116
STATUS	79	None	117
SMFWTM	83	DSSPATCH	118
(IGC084)	84	TESTAUTH	119
SWAP	85	GETMAIN/FREEMAIN	120
EMSERV	89	None	121
VOLSTAT	91	None	122
TPUT/TGET	93	PURGEDQ	123
TSO terminal control	94	TPIO	124
SYSEVENT	95		

Basic SVC Comprehensive Trace Record with Parameter List Information: For detailed information about data gathered for the following SVCs, see *z/OS MVS Diagnosis: Reference*.

Name	Number	Name	Number
EXCP	0	STIMER	47
WAIT/WAITR	1	DEQ	48
POST	2	SNAP	51
GETMAIN	4	RESTART	52
FREEMAIN	5	RELEX	53
LINK	6	DISABLE	54
XCTL	7	EOV	55
LOAD	8	ENQ/RESERVE	56
DELETE	9	FREEDBUF	57
ABEND	13	RELBUF/REQBUF	58
SPIE	14	STAE	60
ERREXCP	15	DETACH	62
PURGE	16	CHKPT	63
RESTORE	17	RDJFCB	64
BLDL/FIND	18	BTAMTEST	66
OPEN	19	BSP	69
CLOSE	20	GSERV	70
STOW	21	ASGNBFR/BUFINQ/ RLSEBFR	71
OPEN TYPE = J	22	SPAR	73
CLOSE TYPE = T	23	DAR	74
DEVTYPE	24	DQUEUE	75
TRKBAL	25	LSPACE	78
CATLG	26	GJP	80
OBTAIN	27	SETPRT	81
CVOL	28	DISKANAL	82
SCRATCH	29	ATLAS	86
RENAME	30	DOM	87
FEOB	31	MOD88	88
ALLOC	32	TCBEXCP	92
WTO/WROR	35	PROTECT	98
SEGLD/SEGWT	37	Dynamic allocation	99
LABEL	39	TCAM	102
EXTRACT	40	EXCPVR	114

Name	Number	Name	Number
IDENTIFY	41		
ATTACH	42		
CHAP	44		
OVLYBRCH	45		

SVC Minimal Trace Record

GTF builds an SVC minimal record when an SVC interruption occurs and TRACE=SYSM is the GTF option in effect.

The FID for the SVC minimal trace record is X'010'. The EID is X'1000'.

Offset	Size	Description
0 (0)	4	ASCB.
4 (4)	2	CPUID.
6 (6)	8	SVC old PSW. The third and fourth bytes contain the SVC number.
14 (E)	4	Old TCB.
18 (12)	4	Register 15.
22 (16)	4	Register 0.
26 (1A)	4	Register 1.

Chapter 11. Component Trace

Component trace is like a Swiss Army Knife: a lot of little tools built into one.

The component trace service provides a way for MVS components to collect problem data about events. Each component that uses the component trace service has set up its trace in a way that provides the unique data needed for the component.

A component trace provides data about events that occur in the component. The trace data is intended for the IBM Support Center, which can use the trace to:
- Diagnose problems in the component
- See how the component is running

You will typically use component trace while re-creating a problem.

Usage of System Resources: Some component traces use minimal system resources, especially while tracing a small number of events. These minimal traces often run anytime the component is running. Other traces use significant system resources, especially when many kinds of events are being traced. These large traces should be requested only when the IBM Support Center asks for them.

Running Concurrent Traces: You can run more than one component trace at a time; you can run component traces:
- Concurrently for several components on one system.
- Concurrently for one or more components on some or all of the systems in a sysplex.
- Concurrently for one component on a system. Multiple concurrent traces for a component are called ***sublevel traces***.
- Concurrently for several components on some or all of the systems in a sysplex and with sublevel traces.

Topics in This Chapter: This chapter describes tasks for component traces:
- “Planning for Component Tracing” on page 11-3 tells you the tasks needed to plan component tracing.
- “Obtaining a Component Trace” on page 11-10 tells you how to request a specific component trace that is needed to diagnose a problem. The tasks depend on where you plan to put trace output and if you are running traces on multiple systems in a sysplex; therefore, requesting traces is presented in three topics:
  - “Request Component Tracing to Address-Space or Data-Space Trace Buffers” on page 11-10
  - “Request Writing Component Trace Data to Trace Data Sets” on page 11-13
  - “Request Component Tracing for Systems in a Sysplex” on page 11-18
- “Verifying Component Tracing” on page 11-21 tells how an operator can check that a requested trace is running and that a component trace writer is running.
- “Viewing the Component Trace Data” on page 11-23 tells you how to format the trace output.

This chapter uses tables to show the similarities and differences in the individual traces from different components. The chapter also describes each BCP component trace that uses the component trace service:

Component	Trace Name	See Topic
Advanced Program-to-Program Communication/MVS (APPC/MVS)	SYSAPPC	11-25
Cross-system coupling facility (XCF)	SYSXCF	11-127
Cross-system extended services (XES)	SYSXES	11-131
Data lookaside facility (DLF) of VLF	SYSDLF	11-39
Distributed function of SOMobjects®	SYSDSOM	11-41
Global resource serialization	SYSGRS	11-43
Allocation Component	SYSIEFAL	11-49
IOS Component Trace	SYSIOS	11-54
JES common coupling services	SYSJES	11-60
JES2 rolling trace table	SYS*jes2*	11-71
Library lookaside (LLA) of contents supervision	SYSLLA	11-73
z/OS UNIX® System Services (z/OS UNIX)	SYSOMVS	11-81
Operations services (OPS)	SYSOPS	11-93
Resource recovery services (RRS)	SYSRRS	11-97
Real storage manager (RSM)	SYSRSM	11-105
Service processor interface (SPI)	SYSSPI	11-120
System logger	SYSLOGR	11-74
Transaction trace (TTRC)	SYSTTRC	11-121
Virtual lookaside facility (VLF)	SYSVLF	11-121
Workload manager (WLM)	SYSWLM	11-125

A program product or application, if authorized, can also use the component trace service to provide an ***application trace***. See the documentation for the program product or application for information about its trace.

References

- See *z/OS MVS Initialization and Tuning Reference* for the CTncccxx parmlib member.
- See *z/OS MVS System Commands* for the TRACE CT® command.
- See *z/OS MVS IPCS Commands* for the COPYDUMP, COPYTRC, CTRACE, GTFTRACE, and MERGE subcommands.
- For a description of these messages, use LookAt or see *MVS System Messages*. For a description of LookAt, see "Using LookAt to look up message explanations" on page xviii.

- See *z/OS MVS Programming: Authorized Assembler Services Guide* for information on creating an ***application trace***.

Planning for Component Tracing

Planning for component tracing consists of the following tasks:
- “Create CTnccсxx Parmlib Members for Some Components”:
  - “Specify Buffers” on page 11-5
- “Select the Trace Options for the Component Trace” on page 11-8
- “Decide Where to Collect the Trace Records” on page 11-9

The system programmer performs the tasks.

Create CTncccxx Parmlib Members for Some Components

The following table shows if a component has a parmlib member, if the member is a default member needed at system or component initialization, and if the component has default tracing. Some components run default tracing at all times when the component is running; default tracing is usually minimal and covers only unexpected events. Other components run traces only when requested.

When preparing your production SYS1.PARMLIB system library, do the following:

1. Make sure the parmlib contains all default members identified in the table. If parmlib does not contain the default members at initialization, the system issues messages.

   Make sure that the IBM-supplied CTIITT00 member is in the parmlib. PARM=CTIITT00 can be specified on a TRACE CT command for a component trace that does not have a parmlib member; CTIITT00 prevents the system from prompting for a REPLY after the TRACE CT command. In a sysplex, CTIITT00 is useful to prevent each system from requesting a reply.
2. Decide if each default member meets the needs of your installation. If it does not, customize it.
3. Decide if the buffer size specified in the default members meets the needs of your installation. For some component traces, the buffer size cannot be changed after initialization. Change the buffer size, if needed.

Most components can run only one component trace at a time; some components can run concurrent traces, called ***sublevel traces***. Each sublevel trace is identified by its sublevel trace name.

Trace	Parmlib Member	Default Member	Default Tracing Beginning at Initialization	Sublevel Traces
SYSAPPC	CTnAPPxx See “CTnAPPxx Parmlib Member” on page 11-26.	No	No; cannot turn trace ON or OFF in CTnAPPxx	No
SYSDLF	None	N/A	Yes; always on when DLF is running	No
SYSDSOM	None	N/A	No	Yes
SYSGRS	CTnGRSxx See “CTnGRSxx Parmlib Member” on page 11-44.	CTIGRS00, which is specified in GRSCNF00 member	Yes, if global resource serialization is active; CONTROL and MONITOR options	No

Component Trace

Trace	Parmlib Member	Default Member	Default Tracing Beginning at Initialization	Sublevel Traces
SYSIEFAL	CTIIEFxx See "CTIIEFxx Parmlib Member" on page 11-49.	CTIIEFAL	Yes	No
SYSIOS	CTnIOSxx See "CTnIOSxx Parmlib Member" on page 11-55.	No	Yes; minimal; unexpected events	No
SYSJES	CTnJESxx See "CTnJESxx Parmlib Member" on page 11-62.	CTIJES01, CTIJES02, CTIJES03, CTIJES04 You should also receive and rename members IXZCTION and IXZCTIOF supplied in SYS1.SAMPLIB to CTIJESON and CTIJESOF.	Yes; full tracing for sublevels XCFEVT and FLOW; minimal tracing of unexpected events for sublevels USRXIT and MSGTRC	Yes
SYS*jes2*	None	N/A	Yes; always on when JES2 is running	Yes
SYSLLA	None	N/A	Yes; always on when LLA is running	No
SYSLOGR	CTnLOGxx See "CTnLOGxx Parmlib Member" on page 11-77.	No	Yes; minimal; unexpected events	No
SYSOMVS	CTnBPXxx See "CTnBPXxx Parmlib Member" on page 11-82.	CTIBPX00, which must be specified in BPXPRM00 member	Yes; minimal; unexpected events	No
SYSOPS	CTnOPSxx See "CTnOPSxx Parmlib Member" on page 11-93.	CTIOPS00, which must be specified in CONSOLxx member	Yes; minimal; unexpected events	No
SYSRRS	CTnRRSxx See "CTnRRSxx Parmlib Member" on page 11-98.	No	Yes; minimal; unexpected events	No
SYSRSM	CTnRSMxx See "CTnRSMxx Parmlib Member" on page 11-106.	No	No	No
SYSSPI	None	N/A	No	No
SYSTTRC	N/A	N/A	No	No

Trace	Parmlib Member	Default Member	Default Tracing Beginning at Initialization	Sublevel Traces
SYSVLF	None	N/A	Yes; minimal; unexpected events	No
SYSWLM	None	N/A	Yes; minimal; unexpected events	No
SYSXCF	CTnXCFxx See "CTnXCFxx Parmlib Member" on page 11-128.	CTIXCF00, which can be specified in COUPLE00 member	Yes; minimal; unexpected events	No
SYSXES	CTnXESxx See "CTnXESxx Parmlib Member" on page 11-133.	CTIXES00, which can be specified in COUPLE00 member	Yes; minimal; unexpected events	Yes

For some components, you need to identify the component's CTncccxx member in another parmlib member; the components with this requirement have the other parmlib member listed in the default member column of the preceding table.

Example: Specifying the CTncccxx Member

For XCF, specify CTIXCF00 on the CTRACE parameter in the COUPLExx parmlib member.

```
COUPLE SYSPLEX( ...
     CTRACE(CTIXCF00)
     ...
```

Specify Buffers

Each component determines the buffer size and how it is specified. Depending on the component, you may or may not be able to change the buffer size. You may be able to change the size only at system or component initialization, or when the trace is started, or at any time, including while the trace is running. The following table shows how the buffers are specified.

The buffer size determines whether you get all the records needed for diagnosis; when the buffer is full, the system wraps the buffer, overwriting the oldest records. To change the size of the buffer, specify an nnnnK or nnnnM parameter on the TRACE CT operator command or a BUFSIZE parameter in the parmlib member.

Usually you should increase the size of the trace buffer when you increase the amount of tracing. However, if you plan to place a component's trace records in a trace data set, you can probably leave the buffer at its original size. Many component traces do not allow you to change the buffer size after initialization; the table indicates those component traces. If you increase the amount of tracing for one of these traces, specify use of a trace data set, if the component supports its use.

Component Trace

Trace	Default Size and Size Range	Size Set By	Change Size after IPL	Buffer Location
SYSAPPC	512KB 64KB - 32MB	CTnAPPxx member or REPLY for TRACE CT command	Yes, while a trace is running	Data space. A TRACE CT,OFF command requests a dump, which includes the trace buffers.
SYSDLF	N/A	MVS system	No	Data space. In the REPLY for the DUMP command, specify DSPNAME= ('DLF'.CCOFGSDO)
SYSDSOM	N/A	MVS system	No	Private address space
SYSGRS	128KB 128KB - 16MB (System rounds size up to nearest 64KB boundary.)	CTnGRSxx member	Yes.	In the component address space
SYSIEFAL	4M 16KB- 8MB	CTIIEFxx member	Yes.	In the component address space
SYSIOS	36KB 36KB - 1.5MB	CTnIOSxx member or REPLY for TRACE CT command	Yes	Data space. Extended system queue area (ESQA).
SYSJES	N/A	MVS system	No	In the component address space
SYS*jes2*	N/A	JES2	No	In the component address space
SYSLLA	N/A	MVS system	No	In the component address space
SYSLOGR	2MB 2MB - 2047MB	MVS system, CTnLOGxx member, or REPLY for TRACE CT command.	Yes	Data space. In the REPLY for the DUMP command, specify DSPNAME= ('IXLOGR'.*). See "Getting a Dump of System Logger Information" on page 11-75.
SYSOMVS	128K 128KB - 1MB	CTxBPXxx member or REPLY for TRACE CT command	Yes, when initializing z/OS UNIX.	Data space. In the REPLY for the DUMP command, specify DSPNAME= (asid.SYSZBPX1) where asid is the ASID for z/OS UNIX.
SYSOPS	64KB 64KB - 16MB	CTnOPSxx member or REPLY for TRACE CT command	Yes, while a trace is running	Console address space (private).

Trace	Default Size and Size Range	Size Set By	Change Size after IPL	Buffer Location
SYSRRS	1MB 1MB - 2045MB	CTxRRSxx member or REPLY for TRACE CT command	Yes, when restarting a trace after stopping it	Data space and component address space. In the REPLY for the DUMP command, specify DSPNAME=('RRS' .ATRTRACE) and SDATA=RGN.
SYSRSM	3 buffers of 32 pages 2 - 7 address-space buffers, 4- 262,144 pages per buffer 1 - 2047MB for data-space buffers	CTnRSMxx member or REPLY for TRACE CT command	Yes, when starting a trace	Common service area (CSA) or, if specified in CTnRSMxx, a data space. Message IAR007I provides the data space name.
SYSSPI	64KB	MVS system	Yes, when starting a trace	In the component address space
SYSTTRC	1 MB 16K - 999K 1MB - 32MB	MVS system	Yes	Data space owned by the system trace address space
SYSVLF	N/A	MVS system	No	Data space. Enter DISPLAY J,VLF to identify the VLF data spaces. In the REPLY for the DUMP command, specify DSPNAME=('VLF'.Dclsname, 'VLF'.Cclsname), where *clsname* is a VLF class name.
SYSWLM	64KB 64KB - 16M	MVS system	Yes, when starting a trace	Extended common service area (ECSA)
SYSXCF	72KB 16KB - 16MB (System rounds size up to a multiple of 72 bytes.)	CTnXCFxx member	No	Extended local system queue area (ELSQA) of the XCF address space
SYSXES	168KB 16KB - 16MB	CTnXESxx member or REPLY for TRACE CT command	Yes, while a trace is running.	Data space. In the REPLY for the DUMP command, specify SDATA=XESDATA and DSPNAME=(*asid*.IXLCTCAD) where *asid* is the ASID for address space XCFAS

Example: CTIXCF00 Parmlib Member Used at IPL

For XCF, the default buffer size is 72KB. Because this size is adequate only for XCF minimum tracing, double that size to 144KB to allow for one or two options, in case additional XCF component tracing is needed. Specify this doubled size in the CTIXCF00 parmlib member. CTIXCF00 turns on minimal XCF tracing.

```
TRACEOPTS
   ON
   BUFSIZE(144K)
```

Select the Trace Options for the Component Trace

If the IBM Support Center requests a trace, the Center might specify the options, if the component trace uses an OPTIONS parameter in its parmlib member or REPLY for the TRACE CT command. The options are:

Trace	Trace Request OPTIONS Parameter.
SYSAPPC	See “OPTIONS Parameter” on page 11-26
SYSDLF	None
SYSDSOM	None
SYSGRS	See “OPTIONS Parameter” on page 11-45
SYSIEFAL	See “OPTIONS Parameter” on page 11-50
SYSIOS	See “OPTIONS Parameter” on page 11-56
SYSJES	None
SYS*jes2*	None
SYSLLA	None
SYSLOGR	See “OPTIONS Parameter” on page 11-78
SYSOMVS	See “OPTIONS Parameter” on page 11-83
SYSOPS	See “OPTIONS Parameter” on page 11-94
SYSRRS	See “OPTIONS Parameter” on page 11-99
SYSRSM	See “OPTIONS Parameter” on page 11-107
SYSTTRC	None
SYSSPI	None
SYSVLF	None
SYSWLM	None
SYSXCF	See “OPTIONS Parameter” on page 11-129
SYSXES	See “OPTIONS Parameter” on page 11-134

You must specify all options you would like to have in effect when you start a trace. Options specified for a previous trace of the same component do not continue to be in effect when the trace is started again.

If the component has default tracing started at initialization by a parmlib member without an OPTIONS parameter, you can return to the default by doing one of the following:

- Stopping the tracing with a TRACE CT,OFF command.

- Specifying OPTIONS() in the REPLY for the TRACE CT command or in the CTncccxx member.

Example: XCF Trace Options

For XCF, the IBM Support Center identifies the options needed to diagnose a particular problem as both of the following:

```
SERIAL
STATUS
```

Decide Where to Collect the Trace Records

Depending on the component, the potential locations of the trace data are:
- In address-space buffers, which are obtained in a dump
- In data-space buffers, which are obtained in a dump
- In a trace data set or sets, if supported by the component trace

Component	Address-Space Buffer	Data-Space Buffer	Trace Data Set
SYSAPPC	No	Yes	No
SYSDLF	Yes	Yes	No
SYSDSOM	Yes	No	Yes
SYSGRS	Yes	No	Yes
SYSIEFAL	Yes	No	Yes
SYSIOS	Yes	Yes	Yes
SYSJES	Yes	No	Yes
SYS*jes2*	Yes	No	No
SYSLLA	Yes	No	No
SYSLOGR	Yes	Yes	Yes
SYSOMVS	No	Yes	Yes
SYSOPS	Yes	No	Yes
SYSRRS	Yes	Yes	Yes
SYSRSM	Yes	Yes	Yes
SYSTTRC	No	Yes	No
SYSSPI	Yes	No	No
SYSVLF	Yes	Yes	No
SYSWLM	Yes	No	Yes
SYSXCF	Yes	No	Yes
SYSXES	No	Yes	Yes

If the trace records of the trace you want to run can be placed in more than one location, you need to select the location. For a component that supports trace data sets, you should choose trace data sets for the following reasons:
- Because you expect a large number of trace records
- To avoid interrupting processing with a dump of the trace data
- To keep the buffer size from limiting the amount of trace data
- To avoid increasing the buffer size

Depending on the component, you might also want to dump the address-space trace buffers and data-space trace buffers.

Note: You may need to consider the amount of auxiliary storage required to back data space buffers. In general, most components which use data space buffers establish a small default value less than 500 kilobytes of virtual storage. Some components allow you to specify values up to 2 gigabytes. The SYSIOS component trace uses a default of 512 megabytes for data space buffers. You should consider SYSIOS and other component data space buffers to ensure that the potential cumulative effect of all CTRACE data space buffers for your system can be accomodated by the local page data sets that you have allocated. For more information on auxiliary storage, refer to *z/OS MVS Initialization and Tuning Guide*.

Obtaining a Component Trace

To obtain a specific component trace, use one of the following procedures:
- “Request Component Tracing to Address-Space or Data-Space Trace Buffers”
- “Request Writing Component Trace Data to Trace Data Sets” on page 11-13
- “Request Component Tracing for Systems in a Sysplex” on page 11-18

Request Component Tracing to Address-Space or Data-Space Trace Buffers

This topic describes how to obtain component trace records in dumps. The trace records are in address-space or data-space trace buffers. The topic contains:
- “Prepare for a Specific Component Trace to Trace Buffers”
- “Perform Component Tracing to Trace Buffers” on page 11-12

Prepare for a Specific Component Trace to Trace Buffers

The system programmer performs the tasks.

1. Select How the Operator Is to Request the Trace: For most component traces, the request can be made by:
- A TRACE CT operator command without a PARM parameter, followed by a reply containing the options
- A TRACE CT operator command with a PARM parameter that specifies a CTncccxx parmlib member containing the options

If you do not use a parmlib member, tell the operator the options.

2. Create a Parmlib Member, if Used: If you use a parmlib member, create the member and place it on SYS1.PARMLIB. Use a parmlib member if the options are complicated and you have access to the SYS1.PARMLIB data set, or if a parmlib member is required by the component, or if you had already set up a parmlib member with the needed options. Use a REPLY for simple options.

See “Create CTncccxx Parmlib Members for Some Components” on page 11-3.

Example: CTWXCF03 Parmlib Member

For XCF, create CTWXCF03 to specify the options.

```
TRACEOPTS
   ON
   OPTIONS('SERIAL','STATUS')
```

3. Determine the Dump to be Used to Obtain the Trace Records: The following table shows how SVC dumps are requested for the component traces. Possible ways of requesting SVC dumps are:

- By a DUMP operator command
- By a SLIP trap
- By the component

For the following failures, other dumps are used:

- Failure of an application program or program product: The program requests a SYSMDUMP dump.
- The system waits, hangs, or enters a loop: The operator requests a stand-alone dump.

Trace	**Request of SVC Dump**
SYSAPPC	By the component when the operator stops SYSAPPC tracing with a TRACE CT,OFF command
SYSDLF	By DUMP or SLIP command
SYSDSOM	By DUMP or SLIP command
SYSGRS	By DUMP or SLIP command
SYSIEFAL	By DUMP or SLIP command
SYSIOS	By DUMP or SLIP command, or by the component • In the REPLY for the DUMP command, specify the IOS address space to be dumped
SYSJES	By the component
SYS*jes2*	By DUMP or SLIP command or component
SYSLLA	By the component
SYSLOGR	By DUMP or SLIP command
SYSOMVS	By DUMP or SLIP command
SYSOPS	By DUMP or SLIP command
SYSRRS	By DUMP or SLIP command
SYSRSM	• By DUMP or SLIP command • Through the DMPREC option on the CTnRSMxx parmlib member or on the REPLY for the TRACE CT command when RSM enters recovery processing (default) • Through the DMPOFF option of CTnRSMxx or the TRACE CT reply when SYSRSM tracing is turned off
SYSTTRC	Automatically dumped by the Tailored SVC Dump Exits function
SYSSPI	By the component
SYSVLF	By DUMP or SLIP command or when SYSVLF full tracing is turned off
SYSWLM	By DUMP or SLIP command

Trace	Request of SVC Dump
SYSXCF	By DUMP or SLIP command
SYSXES	By DUMP or SLIP command

4. Make Sure the Component Trace Buffers Will Be Dumped: The location of the address-space and data-space buffers depends on the component being traced. See the table in “Specify Buffers” on page 11-5 for the location of the buffers. When the component being traced requests an SVC dump, the dump will contain the address-space and/or data-space trace buffers.

Perform Component Tracing to Trace Buffers

The operator performs the tasks. Note that these tasks are for a specific component trace, rather than for a trace started by the system at initialization.

1. Start the Component Trace: The operator enters a TRACE operator command on the console with MVS master authority. The operator replies with the options that you specified.

Example: TRACE CT Command Not Specifying a Parmlib Member

```
trace ct,on,comp=sysxcf
* 21 ITT006A ....
r 21,options=(serial,status),end
```

Example: TRACE CT Command Specifying a Parmlib Member

This example requests the same trace using parmlib member CTWXCF03. When TRACE CT specifies a parmlib member, the system does not issue message ITT006A.

```
trace ct,on,comp=sysxcf,parm=ctwxcf03
```

2. Verify that the Trace Is Running: See “Verifying Component Tracing” on page 11-21.

3. Obtain the Dump Containing the Component Trace Records: The operator obtains the dump that contains the component trace records: an SVC dump, a stand-alone dump, or a dump requested by the component when a problem occurs or when the operator stops the tracing.

Example: DUMP Command and Responses

The example shows a DUMP operator command entered on the console with MVS master authority. The system issues message IEE094D in response to the DUMP command. If you requested, the operator enters dump options in the reply to IEE094D. SDATA options are needed to obtain the trace buffers An address space identifier (ASID) should be specified for the XCF address space; in the example, XCF is in address space 6.

```
dump  comm=(dump for xcf component trace)
* 32 IEE094D ...
r 32,sdata=(couple,sqa,lsqa),asid=6,end
 .
 .
 .
IEA911E ...
```

The system identifies the data set containing the dump in message IEA911E. If an installation exit moves the dump, the operator should look for a message identifying the data set containing the moved dump and tell you the name of the dump and the data set containing it.

If desired, the operator can request more than one dump while a component trace is running.

4. Stop the Component Trace: The operator enters a TRACE CT,OFF command on the console with MVS master authority. For some component traces, the command requests a dump, which contains the trace records.

Example: TRACE CT,OFF Command

```
trace ct,off,comp=sysxcf
```

Request Writing Component Trace Data to Trace Data Sets

This topic describes only the component traces that can be written to trace data sets. The topic contains:
- “Prepare for a Specific Component Trace to Trace Data Sets”
- “Perform Component Tracing to Trace Data Sets” on page 11-16

You can also change the trace data sets being used without stopping the trace. See “Change Trace Data Sets” on page 11-17.

Prepare for a Specific Component Trace to Trace Data Sets

The system programmer performs the following tasks

1. Determine the dispatching priority required for the external writer started task, the server address space for the component’s trace:

While component trace runs under the master scheduler address space, you need to verify that the priority of the external writer is at least equal to, and preferably greater than the priority of the component being traced. For example, if you are tracing COMP(SYSXES) for JOBNAME(IRLMA), the dispatching priority of the external writer should be equal to or greater than that assigned to IRLMA. See *z/OS MVS Initialization and Tuning Guide* for more information on setting priorities.

2. Select How the Operator Is to Request the Trace:

For most component traces, the request can be made by:
- A TRACE CT operator command without a PARM parameter, followed by a reply containing the options
- A TRACE CT operator command with a PARM parameter that specifies a CTncccxx parmlib member containing the options

If you do not use a parmlib member, tell the operator the options.

3. Create Source JCL for the External Writer:

Create source JCL to invoke an external writer, which will send the component trace output to one or more trace data sets. Add a procedure to the SYS1.PROCLIB system library or a job as a member of the data set in the IEFJOBS or IEFPDSI concatenation.

An external writer is not specific for a component but can be used by any application. So you can use the same source JCL again for other tracing later, if needed.

Concurrent traces for different components must use separate source JCL to place their traces in separate data sets.

Because the writer source JCL specifies data sets, use a different set of source JCL for each system in a sysplex. Several systems cannot share the same data set; attempting to share the same data set will lead to contention problems. If your sysplex uses a common SYS1.PROCLIB, you need to specify a unique writer procedure for each system or use a unique job as the source JCL, when tracing the same component on several systems.

Example: Cataloged Procedure for an External Writer

The procedure places trace data on two DASD data sets. The procedure is placed in member WTRD2 of SYS1.PROCLIB.

```
//WTDASD2  PROC
//IEFPROC  EXEC PGM=ITTTRCWR,REGION=32M
//TRCOUT01 DD  DSNAME=SYS1.CTRACE1,VOL=SER=TRACE6,UNIT=DASD,
//         SPACE=(CYL,10),DISP=(NEW,KEEP),DSORG=PS
//TRCOUT02 DD  DSNAME=SYS1.CTRACE2,VOL=SER=TRACE7,UNIT=DASD,
//         SPACE=(CYL,10),DISP=(NEW,KEEP),DSORG=PS
```

Rules for the Source JCL for an External Writer:

- The name specified on the TRACE CT command or CTncccxx parmlib member is the member name of the source JCL; in the preceding example, WTRD2. The name is 1 to 7 alphanumeric or national characters, with the first character alphabetic or national. National characters are represented by the following hexadecimal codes:

Code	U.S. English EBCDIC Character
X'5B'	$
X'7B'	#
X'7C'	@

In other languages, the codes represent different characters.

- The procedure must invoke the external writer program ITTTRCWR. Code the REGION= keyword on the EXEC statement to specify the maximum storage size required by the external writer.
- The source JCL can specify up to 16 trace data sets. The DD statements have ddnames of TRCOUTxx, where xx is 01 through 16.
- The trace data sets must be sequential data sets.
- To help you manage the trace data sets, establish a naming convention so that the data set name indicates the component trace, the date, and so on.
- All of the data sets must be on DASD or tape. Do not mix device classes, such as tape and DASD.

  Within a device class, IBM recommends that you do not mix several types of devices, such as 3380 and 3390. In a mix of device types, the system would use the smallest block size for all the data sets.
- Do **not** specify the following DCB parameters:
  - BLKSIZE. The system uses the optimal block size, which is 4096 or larger.
  - RECFM. The system uses VB.
  - LRECL. The system uses BLKSIZE minus 4.
- Do **not** specify DISP=SHR.
- Do **not** concatenate trace data sets.
- Use a separate member for each component's trace, even though you *can* connect more than one trace to the same member.
- Use the same member for all the sublevel traces for a component. This approach reduces the number of data sets you must manage.
- Use a separate member for each system's component trace, when a component trace runs in two or more of the systems of a sysplex. If the component traces specify the same cataloged procedure in a shared SYS1.PROCLIB, they will use the same data set or group of data sets; in this case, contention might develop for the data set or sets.
- System security may require that you have RACF SYSHIGH authority to access the trace data sets.

Wrapping Trace Data Sets: If the WTRSTART parameter on the CTncccxx parmlib member or TRACE CT operator command specifies:

- WRAP or omits the parameter: When the system reaches the end of the data set or group of data sets, it writes over the oldest data at the start of the data set or first data set.

  The system also uses only the primary extent or extents for the data set or sets. To obtain the maximum degree of control over the number of trace entries for which space will be allocated, specify space allocation in units of the BLKSIZE of your trace data set, no secondary space, and use the option for contiguous allocation. For example, if your BLKSIZE is 8192, code the SPACE keyword as follows:

  ```
  SPACE=(8192,(500,0),,CONTIG)
  ```
- NOWRAP: When the data set or sets are full, the system stops writing trace records to the data set or sets. The system continues writing trace records in the address-space buffers.

  The system also uses the primary and secondary extents of the data set or sets.

Multiple Trace Data Sets: Use multiple data sets to capture all the trace records, even during spikes of activity. For a SYSRSM trace, which typically produces large numbers of trace records, use multiple data sets to keep from losing records. Multiple trace data sets using different DASD devices can improve performance. To view the trace records in chronological sequence, the system programmer can:

- Combine the trace records into one data set, using an IPCS COPYTRC subcommand, then use the CTRACE subcommand to format the records from the data set.
- Use an IPCS MERGE subcommand to format the records from multiple data sets.

The system places component trace records into each trace data set in sequence. For example, for three data sets, the system places:

- Record 1 into data set 1
- Record 2 into data set 2
- Record 3 into data set 3
- Record 4 into data set 1
- Record 5 into data set 2
- And so on

Lost Trace Data: Ctrace will give an indication in the next successfully written record of any trace data that did not reach the output medium. If no further records are written the following message is displayed when the external writer is stopped:

ITT120I SOME CTRACE DATA HAS BEEN LOST.
LAST *nnn* BUFFERS NOT WRITTEN.

Create a Parmlib Member

If you use a parmlib member, create the member and place it on SYS1.PARMLIB. Use a parmlib member if the options are complicated and you have access to the SYS1.PARMLIB data set, or if a parmlib member is required by the component, or if you had already set up a parmlib member with the needed options. Use a REPLY for simple options.

See "Create CTncccxx Parmlib Members for Some Components" on page 11-3.

Example: CTWXCF04 Parmlib Member

For XCF, create CTWXCF04. Notice the two statements for the writer; the WTRSTART statement starts the writer and the WTR statement connects the writer to the component.

```
TRACEOPTS
   WTRSTART(WTDASD2)
   ON
   WTR(WTDASD2)
   OPTIONS('SERIAL','STATUS')
```

Perform Component Tracing to Trace Data Sets

The operator performs the tasks. Note that these tasks are for a specific component trace, rather than for a trace started by the system at initialization.

1. Start the Writer and Component Trace: The operator enters TRACE operator commands on the console with MVS master authority and replies with the options specified by the system programmer.

Example: TRACE CT Command Not Specifying a Parmlib Member

The second TRACE CT command starts the SYSXCF trace; the trace options were selected in a previous example. Notice the two writer operands; the WTRSTART operand starts the writer and the WTR operand connects the writer to the component.

```
trace ct,wtrstart=wtdasd2
trace ct,on,comp=sysxcf
* 44 ITT006A ....
r 44,wtr=wtdasd2,options=(serial,status),end
```

Example: TRACE CT Command Specifying a Parmlib Member

This example requests the same trace using parmlib member CTWXCF04.

```
trace ct,on,comp=sysxcf,parm=ctwxcf04
```

2. Verify that the Trace and the Writer Are Running: See “Verifying Component Tracing” on page 11-21.

3. Stop the Component Trace: The operator enters a TRACE CT command on the console with MVS master authority.

Example 1: TRACE CT,OFF Command

```
trace ct,off,comp=sysxcf
* 56 ITT006A ....
r 56,end
```

Example 2: TRACE CT Command to Disconnect the Writer

To stop sending trace records to the trace data set, but keep the trace running, the operator can enter the following when the trace is currently running.

```
trace ct,on,comp=sysxcf
* 56 ITT006A ....
r 56,wtr=disconnect,end
```

The operator should stop the external writer.

4. Stop the External Writer: The operator enters a TRACE CT command on the console with MVS master authority.

Example: TRACE CT,WTRSTOP Command

```
trace ct,wtrstop=wtdasd2
```

Change Trace Data Sets

If you are running a component trace to a trace data set or sets, you can determine if you have the needed records without stopping the trace. Ask the operator to do the following:

1. Enter a TRACE CT,WTRSTART command for a different set of source JCL for each external writer to trace data sets.
2. Enter a TRACE CT command that starts the trace with the different source JCL for the writer.

The new source JCL sends the trace records to the new data set or sets. You may lose a few trace records.

You can view the previous data set or sets to check the trace records collected, then continue or stop the trace, as needed.

Example: Changing the Trace Data Sets

The original commands were:

```
trace ct,wtrstart=wtdasd2
trace ct,on,comp=sysxcf
* 67 ITT006A ...
r 67 wtr=wtdasd2,options(serial,status),end
```

The commands to change the trace data sets are:

```
trace ct,wtrstart=wtdasd4
trace ct,on,comp=sysxcf
* 67 ITT006A ...
r 67 wtr=wtdasd4,options(serial,status),end
```

Request Component Tracing for Systems in a Sysplex

This topic describes one way to obtain traces for a component on more than one system in a sysplex. The approach is to obtain a trace in the dump of each system and merge the traces from the dumps, using an IPCS MERGE subcommand. To be useful for diagnosis, the traces must cover the same time period and end at the same time. The topic contains:

- “Prepare for Specific Component Traces on Systems in a Sysplex”
- “Perform Component Tracing on the Systems in the Sysplex” on page 11-19

You can also trace to data sets, if each system uses a unique source JCL for each external writer, so that the trace for each system goes to its own data set. If your installation has a shared SYS1.PROCLIB system library, use a unique parmlib member for each system; each unique parmlib member must specify a unique set of source JCL. If the source JCL is shared, all systems will write trace records on one data set, possibly causing contention problems.

Prepare for Specific Component Traces on Systems in a Sysplex

The system programmer performs the tasks.

1. Create a Parmlib Member to Start the Traces: Create a parmlib member to start the traces of the component. Place the member in the shared SYS1.PARMLIB for the sysplex or in the parmlib for each system to be traced. If a parmlib member is used for each system, give it the same name so that one TRACE CT command can start all the component traces on the systems.

See “Create CTncccxx Parmlib Members for Some Components” on page 11-3.

Example: CTWXCF33 to Start XCF Trace

For XCF, create CTWXCF33 to start the trace.

```
TRACEOPTS
   ON
   OPTIONS('SERIAL','STATUS')
```

The directions for the task assume that a parmlib member can be used. If the component to be traced does not have a parmlib member, the operator can start it with a TRACE CT command in a ROUTE command. The operator has to enter a reply for each system. (The ROUTE command can be used only on MVS systems with JES2.)

2. Make Sure the Component Trace Buffers Will Be Dumped: The location of the address-space and data-space trace buffers depends on the component being traced. For XCF, the extended local system queue area (ELSQA) of the XCF address space contains the XCF component trace buffers. For XES, IXLCTCAD, a data space associated with the XCF address space, contains the XES component trace buffers.

Example: Obtaining XCF and XES Trace Buffers

- For XCF, the operator should specify SQA and LSQA on the REPLY for the DUMP command.
- For XES, the operator should specify SDATA=(XESDATA) and DSPNAME=(*asid*.IXLCTCAD) on the REPLY for the DUMP command, where *asid*'' is XCFAS or 6.

Perform Component Tracing on the Systems in the Sysplex

The operator performs the tasks. Note that these tasks are for a specific component trace, rather than for a trace started by the system at initialization.

1. Start the Component Traces: On a console with MVS master authority on one system in the sysplex, the operator enters a ROUTE command containing a TRACE CT command. (The ROUTE command can be used only on MVS systems with JES2.)

The command specifies a parmlib member with the same name in each system being traced. Note that, if parmlib members are not specified, all systems issue message ITT006A to prompt for options. If the component to be traced does not have a parmlib member, specify the IBM-supplied CTIITT00 member to avoid the prompts.

Example 1: Command to Start Traces in All Systems

The command starts the trace in all systems in the sysplex.

```
route *all,trace ct,on,comp=sysxcf,parm=ctwxcf33
```

Example 2: Command to Start Traces in Some Systems

The command starts the trace in a subset of systems. Both commands specify the CTWXCF33 parmlib member on each system being traced.

```
route subs2,trace ct,on,comp=sysxcf,parm=ctwxcf33
```

Example 3: Command for a Component without a Parmlib Member

The following command turns on tracing for a SYSVLF trace in the systems of a sysplex, without prompts for replies to the TRACE command. The SYSVLF component trace has no parmlib member.

```
route *all,trace ct,on,comp=sysvlf,parm=ctiitt00
```

2. Verify that the Traces Are Running: See "Verifying Component Tracing" on page 11-21.

3. Obtain the Dumps Containing the Component Trace Records: The operator requests an SVC dump for each system being traced.

Example: DUMP Command for Systems in a Sysplex

The example shows the DUMP operator command entered on a console with MVS master authority on one system in the sysplex. The reply requests dumps on all of the systems named in the pattern of S*. The example assumes that the systems being traced have the following names: S1, S2, S3, and S4; any other systems in the sysplex have names that do not fit the pattern, such as B1 or T2.

```
dump  comm=(dump for xcf component trace)
* 32 IEE094D ...
r 32,remote=(syslist=(s*)),end
 .
 .
 .
IEA911E ...
```

The system identifies the data set containing the dump in message IEA911E. If an exit moves a dump, the operator should look for a message identifying the data set containing the moved dump and tell you the name of the dump and the data set containing it.

4. Stop the Component Traces: On a console with MVS master authority on one system in the sysplex, the operator enters a ROUTE command containing a TRACE CT,OFF command to stop the traces. (The ROUTE command can be used only on MVS systems with JES2.)

Example 1: Command to Stop Traces in All Systems

The command stops the traces in all systems in the sysplex.

```
route *all,trace ct,off,comp=sysxcf
```

Example 2: Command to Stop Traces in Some Systems

The command stops the traces in a subset of systems in the sysplex.

```
route subs2,trace ct,off,comp=sysxcf
```

Verifying Component Tracing

The operator should do this task after starting a component trace to make sure that it started successfully. How the task is done depends on whether the component trace has sublevels and whether an external writer is used. This topic contains:

- "Verify Tracing for Component Traces without Sublevels"
- "Verify Tracing for Component Traces with Sublevels"
- "Verify that the Writer Is Active" on page 11-23

Verify Tracing for Component Traces without Sublevels

The operator should do one of the following:

- Identify all current tracing by entering the following DISPLAY TRACE command on the console with MVS master authority. The response, in message IEE843I, gives the status in short form of all current component traces.

  ```
  display trace
  IEE843I ...
  ```

- Identify current tracing and the options for the traces by entering one of the following DISPLAY TRACE commands on the console with MVS master authority. The first command requests the status for all current component traces; the second command requests it for one component trace, such as XCF. The response, in message IEE843I, gives full information about the status.

  ```
  display trace,comp=all
  IEE843I ...

  display trace,comp=sysxcf
  IEE843I ...
  ```

Verify Tracing for Component Traces with Sublevels

The commands for verification depend on the component trace.

Verify a SYSJES Component Trace: The operator enters the following command to verify a SYSJES trace:

```
display trace,comp=sysjes,sublevel,n=4
```

The system will show the 4 sublevel traces.

Verify a SYSXES Component Trace: When a SYSXES component trace has multiple sublevel traces, a DISPLAY command shows only one sublevel. The operator needs to enter multiple DISPLAY commands to see the multiple sublevels.

A SYSXES component trace has structures, address spaces, and connections. The following examples show the DISPLAY (D) command entered by the operator and the type of information that the system returns.

1. To see how the SYSXES component trace is set up.

   ```
   D TRACE,COMP=SYSXES

   IEE843I 15.24.40  TRACE DISPLAY 213
           SYSTEM STATUS INFORMATION
    ST=(ON,0064K,00128K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
     COMPONENT     MODE BUFFER HEAD SUBS
   ```

```
  --------------------------------------------------------
   SYSXES       ON   0168K  HEAD    2
      ASIDS      *NOT SUPPORTED*
      JOBNAMES   *NOT SUPPORTED*
      OPTIONS    LOCKMGR
      WRITER     *NONE*
```

2. To display the structure level trace for each structure and the number of subtraces available.

```
D TRACE,COMP=SYSXES,SUB=(LT01),N=99

IEE843I 15.25.00  TRACE DISPLAY 216
        SYSTEM STATUS INFORMATION
 ST=(ON,0064K,00128K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
  TRACENAME
  =========
  SYSXES
                      MODE BUFFER HEAD SUBS
                      =====================
                      ON   0168K  HEAD    2
     ASIDS      *NOT SUPPORTED*
     JOBNAMES   *NOT SUPPORTED*
     OPTIONS    LOCKMGR
     WRITER     *NONE*
  SUBTRACE            MODE BUFFER HEAD SUBS
 ----------------------------------------------------------
   LT01                            HEAD    1
      LIKEHEAD
  -----------------------------------------------------
   GLOBAL
      LIKEHEAD
```

3. To display the address space level trace for each structure. (The ASID specified is the asid in hex of the address space of the connector.)

```
D TRACE,COMP=SYSXES,SUB=(LT01.ASID(19)),N=99

IEE843I 15.25.39  TRACE DISPLAY 221
        SYSTEM STATUS INFORMATION
 ST=(ON,0064K,00128K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
  TRACENAME
  =========
  SYSXES.LT01
                      MODE BUFFER HEAD SUBS
                      =====================
                      ON   0168K  HEAD    1
     LIKEHEAD
     ASIDS      *NOT SUPPORTED*
     JOBNAMES   *NOT SUPPORTED*
     OPTIONS    LOCKMGR
     WRITER     *NONE*

  SUBTRACE            MODE BUFFER HEAD SUBS
 --------------------------------------------------------------
  ASID(0019)                      HEAD    8
     LIKEHEAD
```

4. To display the external writer and the buffer size and options associated with the connection.

```
D TRACE,COMP=SYSXES,SUB=(LT01.ASID(19).A1),N=99

IEE843I 15.25.56  TRACE DISPLAY 224
        SYSTEM STATUS INFORMATION
 ST=(ON,0064K,00128K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
  TRACENAME
  =========
  SYSXES.LT01.ASID(0019)
                      MODE BUFFER HEAD SUBS
```

```
                     =====================
                     ON   0168K  HEAD    8
    LIKEHEAD
    ASIDS      *NOT SUPPORTED*
    JOBNAMES   *NOT SUPPORTED*
    OPTIONS    LOCKMGR
    WRITER     *NONE*
SUBTRACE            MODE BUFFER HEAD SUBS
--------------------------------------------------------------
A1                  ON   0100K
   ASIDS      *NOT SUPPORTED*
   JOBNAMES   *NOT SUPPORTED*
   OPTIONS    CONNECT,RECOVERY
   WRITER     *NONE*
```

Verify that the Writer Is Active

If an external writer is being used, the operator should verify that the writer is active for the trace by entering one of the following DISPLAY TRACE commands on the console with MVS master authority. The first command requests writer status for all current component traces; the second command requests it for one writer by specifying the membername of the source JCL for the writer, such as WTDASD2. The response is in message IEE843I.

```
display trace,wtr=all
IEE843I ...

display trace,wtr=wtdasd2
IEE843I ...
```

The operator should verify that the source JCL for the writer in this display is the same as the source JCL for the writer that was started for the trace. If the membernames do not match, the component trace data is lost. The operator should stop the writer job identified in the display and the component trace; then start the correct writer source JCL and start the trace again.

Viewing the Component Trace Data

During diagnosis, the system programmer performs the tasks, using IPCS.

Reference

See *z/OS MVS IPCS Commands* for the COPYDUMP, COPYTRC, CTRACE, GTFTRACE, and MERGE subcommands.

1. If your trace is in a dump in a SYS1.DUMPxx data set, enter a COPYDUMP subcommand to move the dump to another data set. Use option 5.3 of the IPCS dialog to select the COPYDUMP subcommand.

2. For all traces on trace data sets, use a COPYTRC subcommand to reorder component trace records that are out of chronological sequence. Use option 5.3 of the IPCS dialog to select the COPYTRC subcommand.

3. If your trace is on multiple data sets, do one of the following to view the trace records in one chronological sequence, which is needed to understand what was happening when the problem occurred. The input data sets can be component trace data sets, SVC dumps, and stand-alone dumps.

- Use the COPYTRC subcommand to combine the records on several data sets into a chronological sequence on one data set. Use this data set as input to the

CTRACE subcommand, which formats the trace records. Use option 5.3 of the IPCS dialog to select the COPYTRC subcommand.

- Use a MERGE subcommand to format trace records from one or more input data sets. MERGE lets you combine and format the following:
  - Component traces
  - GTF traces
  - Sublevel traces from one component on one trace data set
  - Sublevel traces from one component on separate trace data sets

  For sublevel traces, MERGE groups together the trace records for each sublevel.

  Use option 2.7 of the IPCS dialog to select the MERGE subcommand. MERGE allows you to issue individual CTRACE or GTFTRACE subcommands for each input data set.

4. Use the following subcommands when formatting the component trace records. See *z/OS MVS IPCS Commands* for the SHORT, SUMMARY, FULL, and TALLY report type keywords and other keywords for the CTRACE subcommand.

Trace	IPCS subcommand	CTRACE OPTIONS Parameter
SYSAPPC	CTRACE COMP(SYSAPPC)	See “Formatting a SYSAPPC Trace” on page 11-29
SYSDLF	CTRACE COMP(SYSDLF)	None
SYSDSOM	CTRACE COMP(SYSDSOM)	See “SYSDSOM Component Trace” on page 11-41
SYSGRS	CTRACE COMP(SYSGRS)	None
SYSIEFAL	CTRACE COMP(SYSIEFAL)	None
SYSIOS	CTRACE COMP(SYSIOS)	See “Formatting a SYSIOS Trace” on page 11-58
SYSJES	CTRACE COMP(SYSJES)	See “Formatting a SYSJES Trace” on page 11-65
SYS*jes2*	CTRACE COMP(SYS*jes2*)	None
SYSLLA	CTRACE COMP(SYSLLA)	None
SYSLOGR	CTRACE COMP(SYSLOGR)	See “Formatting a SYSLOGR Trace” on page 11-80
SYSOMVS	CTRACE COMP(SYSOMVS)	See “Formatting a SYSOMVS Trace” on page 11-84
SYSOPS	CTRACE COMP(SYSOPS)	See “Formatting a SYSOPS Trace” on page 11-95
SYSRRS	CTRACE COMP(SYSRRS)	See “Formatting a SYSRRS Trace” on page 11-101
SYSRSM	CTRACE COMP(SYSRSM)	None
SYSSPI	CTRACE COMP(SYSSPI)	None
SYSTTRC	CTRACE COMP(SYSTTRC)	None
SYSVLF	CTRACE COMP(SYSVLF)	None
SYSWLM	CTRACE COMP(SYSWLM)	None
SYSXCF	CTRACE COMP(SYSXCF)	See “Formatting a SYSXCF Trace” on page 11-130
SYSXES	CTRACE COMP(SYSXES)	See “Formatting a SYSXES Trace” on page 11-135

If some of the output in a combined or merged trace data set is for a GTF trace, use a GTFTRACE subcommand to format the GTF records and a CTRACE subcommand to format the component trace records.

Reference

See Chapter 10, “The Generalized Trace Facility (GTF)” on page 10-1 for GTF tracing.

Example: IPCS CTRACE Subcommand

The example shows the CTRACE subcommand for a SYSXCF component trace, when the SERIAL and STATUS options are requested in the OPTIONS parameter.

```
ctrace comp(sysxcf) options((serial,status))
```

SYSAPPC Component Trace

Before Using This Component Trace

This topic assumes you have read:
- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSAPPC component trace for APPC/MVS.

Information	For SYSAPPC:
Parmlib member	CTnAPPxx No default member
Default tracing	No; cannot turn trace ON or OFF in CTnAPPxx
Trace request OPTIONS parameter	In CTnAPPxx or REPLY for TRACE command
Buffer	• Default: 512KB • Range: 64KB - 32MB • Size set by: CTnAPPxx member or REPLY for TRACE command • Change size after IPL: Yes, while a trace is running • Location: In data space. A TRACE CT,OFF command requests a dump, which includes the trace buffers.
Trace records location	Data-space buffer
Request of SVC dump	By the component when the operator stops SYSAPPC tracing with a TRACE CT,OFF command
Trace formatting by IPCS	CTRACE COMP(SYSAPPC)
Trace format OPTIONS parameter	Yes

Requesting a SYSAPPC Trace

Specify options for requesting a SYSAPPC component trace on a CTnAPPxx parmlib member or on the reply for a TRACE CT command.

CTnAPPxx Parmlib Member

The following table indicates the parameters you can specify on a CTnAPPxx parmlib member.

Parameters	Allowed on CTnAPPxx?
ON or OFF	No
ASID	Yes
JOBNAME	Yes
BUFSIZE	Yes
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No
WTR	No
WTRSTART or WTRSTOP	No

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON, nnnnK, nnM, or OFF	One is required
COMP	Required
SUB	No
PARM	Yes

Parameters	Allowed on REPLY for Trace?
ASID	Yes
JOBNAME	Yes
OPTIONS	Yes
WTR	No

Automatic Dump: The component requests an SVC dump when the operator stops the trace.

OPTIONS Parameter

APPC trace request options are **hierarchical**. Figure 11-1 on page 11-27 shows the hierarchy of SYSAPPC trace options. Each option traces its own events, plus all the events of the options below it. For example, if you specify the SCHEDULE trace option, the system also traces ENQWORK, DEQWORK, and ASMANAGE events.

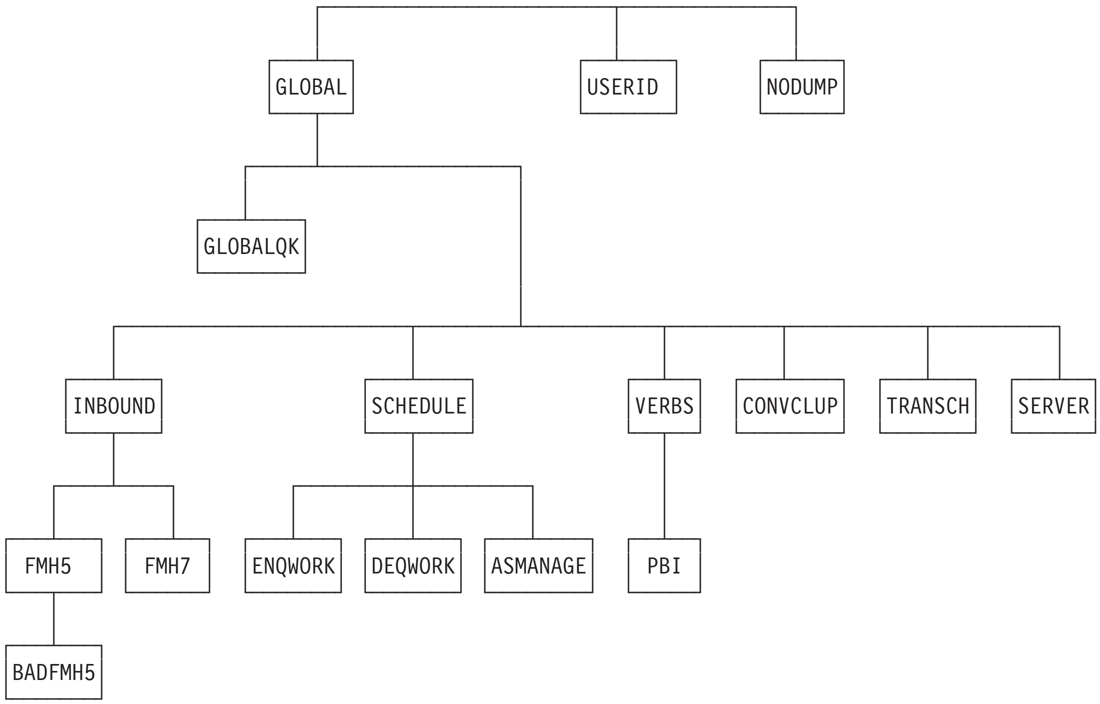

Figure 11-1. Hierarchy of SYSAPPC Component Trace Options

SYSAPPC tracing always includes all exception (error) events. If no trace options are specified, the trace output includes only the exception events.

If you do not know where the errors are occurring, use the GLOBAL trace option to catch the full range of APPC/MVS events. GLOBAL can slow performance, but you will catch the error in one re-create.

The values for the OPTIONS parameter for the CTnAPPxx parmlib member and reply for a TRACE command, in alphabetical order, are:

ASMANAGE
: Traces events related to the creation and deletion of the APPC/MVS transaction scheduler's subordinate address space. ASMANAGE is a subset of SCHEDULE events.

BADFMH5
: Traces events related to incorrect FMH-5s. BADFMH5 is a subset of FMH-5 events.

CONVCLUP
: Traces events related to conversation cleanup. CONVCLUP is a subset of GLOBAL events.

DEQWORK
: Traces the process of removing work requests from an APPC/MVS scheduler queue. DEQWORK is a subset of SCHEDULE events.

ENQWORK
: Traces the process of adding work requests to an APPC/MVS scheduler queue. ENQWORK is a subset of SCHEDULE events.

FMH5
: Traces FMH-5 events. FMH5 is a subset of INBOUND events.

FMH7
Traces FMH-7 events. FMH7 is a subset of INBOUND events.

GLOBAL
Traces the full range of APPC/MVS events.

GLOBALQK
Traces a subset of important GLOBAL trace events.

INBOUND
Traces inbound transaction processor (TP) requests. INBOUND is a subset of GLOBAL events.

NODUMP
Specifies no dumping when the operator stops the SYSAPPC component trace. Otherwise, component trace requests an SVC dump with the trace data when the operator stops tracing with a TRACE CT,OFF command.

IBM does not recommend the NODUMP option because the option makes obtaining the trace buffers difficult. The operator would have to identify the data space containing the buffers and specify it in a SLIP command or the reply for a DUMP command.

PBI
Traces events related to protocol boundary. PBI is a subset of VERBS events.

RR
Traces events related to the participation of APPC/MVS in resource recovery for protected conversations. RR is a subset of VERBS events.

SERVER
Traces events related to the APPC/MVS servers. SERVER is a subset of GLOBAL events.

SCHEDULE
Traces events related to the APPC/MVS transaction scheduler. SCHEDULE is a subset of GLOBAL events.

TRANSCH
Traces events related to APPC/MVS transaction scheduler interface support. TRANSCH is a subset of GLOBAL events.

USERID=(*userid*,*userid*)
Traces events for only the specified userid or userids. Specify the TSO/E userid of the person reporting a problem with an APPC/MVS application. Specify from 1 through 9 userids.

VERBS
Traces events related to outbound TPs or LU services. VERBS is a subset of GLOBAL events.

Examples of Requesting SYSAPPC Traces

Example 1: CTnAPPxx Member

The member requests SERVER and VERBS options for the address space or spaces for the TSO/E userid JOHNDOE.

```
TRACEOPTS
   OPTIONS('SERVER','VERBS','USERID=(JOHNDOE)')
```

Example 2: TRACE Command Specifying a Parmlib Member

The example specifies that options are to be obtained from the parmlib member CTWAPP03.

```
trace ct,on,comp=sysappc,parm=ctwapp03
```

Example 3: TRACE Command with Options Specified in a REPLY

The example requests the same trace as Example 2, but specifies all options in the REPLY.

```
trace ct,on,comp=sysappc
* 15 ITT006A ...
reply 15,options=(server,verbs,userid=(johndoe)),end
```

Example 4: TRACE Command Requesting GLOBAL Options

The example requests GLOBAL options for all address spaces using APPC/MVS.

```
trace ct,on,comp=sysappc
* 14 ITT006A ...
reply 14,options=(global),end
```

Formatting a SYSAPPC Trace

Format the trace with an IPCS CTRACE COMP(SYSAPPC) subcommand. Its OPTIONS parameter specifies the options that select trace records to be formatted. Your formatting options depend to a great extent on the tracing options you requested. Use the options to narrow down the records displayed so that you can more easily locate any errors. If the CTRACE subcommand specifies no options, IPCS displays all the trace records.

The options follow. The first option is either FILTER or CORRELATE, which are mutually exclusive; the first option controls how the other options select the records.

FILTER

The FILTER option selects the trace records that match only one of the specified options. The options that are valid with the FILTER option are:

AQTOKEN
CONVCOR
CONVID
FUNCID
INSTNUM
LUNAME
LUWID
NETNAME
SEQNUM
SESSID
TPIDPRI
TPIDSEC
URID

USERID

The formats of the OPTION parameter with FILTER are:

```
OPTION((FILTER,option))
OPTION((FILTER,option,option, ... ,option))
```

CORRELATE

The CORRELATE option selects the trace records that match a specified option and, for an unspecified option, uses the option's default values. The DEFAULTS keyword defines how default values are found for the unspecified options. The options that are valid with the CORRELATE option are:

AQTOKEN
CONVCOR
CONVID
DEFAULTS
INSTNUM
LUNAME
LUWID
NETNAME
SEQNUM
SESSID
TPIDPRI
TPIDSEC
URID

The formats of the OPTION parameter with CORRELATE are:

```
OPTION((CORRELATE,option))
OPTION((CORRELATE,option,option, ... ,option))
```

AQTOKEN(*allocate-queue-token***)**

Use with either the FILTER or CORRELATE option to specify an allocate queue token. The *allocate-queue-token* is an 8-byte hexadecimal string.

CONVCOR(*conversation-correlator***)**

Use with either the FILTER or CORRELATE option to specify a conversation correlation. The *conversation-correlator* is an 8-byte hexadecimal string.

CONVID(*conversation-id***)**

Use with either the FILTER or CORRELATE option to specify a conversation identifier. The *conversation-id* is a 4-byte hexadecimal string.

||DEFAULTS(NONEANYEXACT)

Use only with the CORRELATE option to specify the values to be used for matching unspecified options.

NONE

Tells component trace to format only the trace records that match one or more of the specified options. NONE is the default.

ANY

Tells component trace to format:

- Trace records matching one or more of the specified options.
- Related trace records that match default values established for the unspecified options. Component trace derives the defaults from the values for unspecified options found in the first records that match **any** of the specified options.

EXACT

Tells component trace to format:

- Trace records matching one or more of the specified options.
- Related trace records matching default values established for the unspecified options. Component trace derives the defaults from the values for unspecified options found in the first records that match **all** of the specified options.

FUNCID(*function-id***)**
Use only with the FILTER option to specify the APPC/MVS subcomponent trace records to format. Specify one *function-id*:

01 Recovery

02 Verb services

03 FMH-5 manager

04 Conversation manager

05 System data file manager (SDFM)

06 VTAM exits

07 LU manager

08 State machine

09 Test enablement

10 APPC/MVS scheduler (ASCH)

11 Transaction scheduler interface

12 Allocate queue services

INSTNUM(*instance-number***)**
Use with either the FILTER or CORRELATE option to specify the instance number for a logical unit of work. The *instance-number* is a 6-byte hexadecimal string.

LUNAME(*local-luname***)**
Use with either the FILTER or CORRELATE option to specify the LU name for the local logical unit of work. The *local-luname* is an 8-byte EBCDIC character string.

LUWID(*logical-unit-of-work-id***)**
Use either the FILTER or CORRELATE option to specify a logical unit of work identifier, which represents the processing a program performs from one sync point to the next. To specify the *logical-unit-of-work-id*, enter the hexadecimal string as it appears in the CTRACE report, without including blank spaces.

NETNAME(*network-name***)**
Use with either the FILTER or CORRELATE option to specify the network name for a logical unit of work. The *network-name* is an 8-byte EBCDIC character string, which is the same as the network-ID portion of a network-qualified LU name.

SEQNUM(*sequence-number***)**
Use with either the FILTER or CORRELATE option to specify the sequence number for a logical unit of work. The *sequence-number* is a 2-byte hexadecimal string.

SESSID(*session-id***)**
Use with either the FILTER or CORRELATE option to specify the session identifier. The *session-id* is an 8-byte hexadecimal string.

TPIDPRI(*tp-id*)
Use with either the FILTER or CORRELATE option to specify the primary TP identifier. The *tp-id* is an 8-byte hexadecimal string.

TPIDSEC(*tp-id*)
Use with either the FILTER or CORRELATE option to specify the secondary TP identifier, which is used for multi-trans TPs. The *tp-id* is an 8-byte hexadecimal string.

URID(*unit-of-recovery-id*)
Use with either the FILTER or CORRELATE option to specify a unit of recovery identifier, which represents part of a TP's processing for a protected conversation. The *unit-of-recovery-id* is a 32-byte hexadecimal string.

USERID(*userid*)
Use only with the FILTER option to specify a userid as a filter. The *userid* is an 8-byte EBCDIC character string.

Examples of Subcommands to Format a SYSAPPC Trace

Example 1: CTRACE Subcommand to View All Trace Entries

To view all the SYSAPPC trace records, enter:

```
CTRACE COMP(SYSAPPC)
```

Example 2: CTRACE Subcommand to View Exception Entries

To format abnormal SYSAPPC events, such as abends or VTAM return codes, enter:

```
CTRACE COMP(SYSAPPC) EXCEPTION
```

Example 3: CTRACE Subcommand for Subcomponent

To format all the records for one APPC/MVS subcomponent, enter the following subcommand. Use this subcommand to locate an error if you have narrowed the problem down to one subcomponent.

```
CTRACE COMP(SYSAPPC) OPTIONS((FILTER,FUNCID(nn)))
```

Example 4: CTRACE Subcommand to View a Userid's Entries

To format all the records for userid JOHNDOE, who is experiencing problems, enter the following subcommand. If you specified the USERID option when requesting the trace, this formatting option is redundant.

```
CTRACE COMP(SYSAPPC) OPTIONS((FILTER,USERID(JOHNDOE)))
```

Output from a SYSAPPC Trace

CTRACE COMP(SYSAPPC) SHORT Subcommand Output

The SHORT parameter shows one line of output for each trace record. An example of SYSAPPC component trace output formatted with the SHORT parameter follows:

```
VEFMH5XT  00004101  155705.182844  VEFMH-5 RECEIVED
VEFMH5ER  00000101  155705.367233  VEFMH-5 IN TPEND
```

The fields in each SHORT report line are:

Mnemonic
For example, VEFMH5XT.

Entry ID
The identifier for the trace record. For example, 00004101.

Time
The time in hh:mm:ss.tttttt format. For example, 15:57:05.182844.

Title
The title of the record. For example, VE:FMH-5 RECEIVED. Each title begins with a prefix that indicates the APPC/MVS subcomponent that wrote the trace record. For example, VE, which represents the VTAM exits subcomponent. The following relates the title prefixes to their APPC/MVS subcomponents.

Prefix	Subcomponent
AMI	Verb services
ASCH	APPC/MVS scheduler (ASCH)
CM	Conversation manager
ERROR	Recovery
FMH5	FMH-5 manager
LUM	LU manager
PC	Protected conversations
SDFM	MVS system data file manager (SDFM)
SF	Allocate queue services
SM	State machine
TE	Test enablement
TSI	Transaction scheduler interface
VC	Verb services
VE	VTAM exits
VS	Verb services

CTRACE COMP(SYSAPPC) SUMMARY Subcommand Output

The SUMMARY parameter gives the line in the SHORT report and most fields in each trace record. An example of SYSAPPC component trace output formatted with the SUMMARY parameter follows:

```
SY1       PCESC     00007802  13:07:29.491950  PC:ENTRY STATE CHECK EXIT
      FUNCID... 02
      USERID... IBMUSER            JOBNAME.. APPC
      ASIDHOME. 001C               ASIDPRI.. 001C
      TPIDPRI.. 00000000  TPIDSEC.. 00000000
      SESSID... E723ED63  AAB04BDF  CONVID... 01000014
      CONVCOR.. 063313F8  0000000D  AQTOKEN.. 00000000
      LUWID.... 10E4E2C9  C2D4E9F0  4BE9F0C3  F0C1D7F0  F36FDB2A  C0220700  01
        NETNAME.. USIBMZ0            LUNAME... Z0C0AP03
        INSTNUM.. 6FDB2AC0  2207      SEQNUM... 0001
      URID..... AD355FDB  7EEFB000  00000007  01010000
```

The fields in the SUMMARY report, after the first line, follow. See the SHORT report for the first line.

FUNCID
An identifier of the APPC/MVS subcomponent that wrote the trace record. See the FUNCID option for the identifiers.

USERID
The system was processing work for this userid when the trace record event occurred.

JOBNAME
The name of the job that the system was processing when the trace record event occurred.

ASIDHOME
The address space identifier (ASID) of the primary address space the system was processing when the trace record event occurred.

TPIDPRI
_The TP identifier of a primary TP. (Multitrans TPs have a primary and a secondary TP.)

TPIDSEC
_The TP identifier for a secondary TP. (Multitrans TPs have a primary and a secondary TP.)

SESSID
The identifier for a session.

CONVID
The identifier for a conversation.

AQTOKEN
The identifier for an allocate queue.

LUWID
The identifier for a logical unit of work. The following fields refer to the logical unit of work: If the LUWID is either all zeros or not valid,* the fields contain asterisks ().

NETNAME
The network name for the logical unit of work.

LUNAME
The name of the local LU.

INSTNUM
The instance number for the logical unit of work.

SEQNUM
The sequence number for the logical unit of work.

URID
The identifier for a unit of recovery.

CTRACE COMP(SYSAPPC) FULL Subcommand Output

The FULL parameter gives all the data in the trace records. It contains the line in the SHORT report, the fields in the SUMMARY report, and KEY and ADDR fields. An example of SYSAPPC component trace output formatted with the FULL parameter follows:

```
SY1       PCESC     00007802  13:07:29.491950  PC:ENTRY STATE CHECK EXIT
     FUNCID... 02
     USERID... IBMUSER             JOBNAME.. APPC
     ASIDHOME. 001C                ASIDPRI.. 001C
     TPIDPRI.. 00000000  TPIDSEC.. 00000000
     SESSID... E723ED63  AAB04BDF  CONVID... 01000014
     CONVCOR.. 063313F8  0000000D  AQTOKEN.. 00000000
     LUWID.... 10E4E2C9  C2D4E9F0  4BE9F0C3  F0C1D7F0  F36FDB2A  C0220700  01
       NETNAME.. USIBMZ0             LUNAME... Z0C0AP03
       INSTNUM.. 6FDB2AC0  2207      SEQNUM... 0001
     URID..... AD355FDB  7EEFB000  00000007  01010000
     KEY...... 0015                ADDR..... 066F26DA
         E4E2C9C2  D4E9F04B  E9F0C3F0  C1D7F0F3  | USIBMZ0.Z0C0AP03 |
     KEY...... 001A                ADDR..... 066F26EB
         E4E2C9C2  D4E9F04B  E9F0C3F0  C1D7F0F4  | USIBMZ0.Z0C0AP04 |
     KEY...... 0039                ADDR..... 066F26A8
         E3D9C1D5  D7C1D940                      | TRANPAR          |
     KEY...... 0054                ADDR..... 064162E4
         00000000                                | ....             |
     KEY...... 00A1                ADDR..... 066F2640
         00000000                                | ....             |
     KEY...... 00A3                ADDR..... 055683D4
         00000000                                | ....             |
     KEY...... 00A3                ADDR..... 066F263C
         00000000                                | ....             |
     KEY...... 00A2                ADDR..... 066F263A
         00                                      | .                |
```

See the SHORT report and the SUMMARY report for the fields in the report. IBM might need the KEY and ADDR fields for diagnosis.

FMH-5 Trace Data

FMH-5 trace records contain information useful for tracking TP flow and diagnosing the following types of problems:
- Persistent verification problems
- Password maintenance problems
- APPC/MVS security problems

To obtain FMH-5 data, request the SYSAPPC component trace with an FMH5, INBOUND, or GLOBAL option. To isolate the FMH-5 records in the trace output, enter the following IPCS subcommand:

```
CTRACE COMP(SYSAPPC) OPTIONS((FILTER,FUNCID(03))) FULL
```

Table 11-1 gives the mnemonic and title for each FMH-5 trace record and explains the record. Most of the trace records have FMH-5 itself formatted in KEY field X'0012'.

Reference

See *SNA Network Product Formats* for the format of the FMH-5.

Table 11-1. FMH-5 Trace Entries in the SYSAPPC Component Trace

Mnemonic	Title	Description/Action
FMH5BDSC	FMH5:BAD SECURITY COMBINATION	APPC/MVS found an incorrect security option or security subfields or both. Contact the IBM Support Center.
FMH5ERCV	FMH5:FMH-5 RECEIVE FAILURE	An FMH-5 was not successfully received by the local MVS LU. Contact the IBM Support Center.
FMH5INCD	FMH5:FMH-5 COMMAND IS NOT VALID	APPC/MVS detected an incorrect FMH-5 command. Contact the IBM Support Center.

Table 11-1. FMH-5 Trace Entries in the SYSAPPC Component Trace (continued)

Mnemonic	Title	Description/Action
FMH5LUNA	FMH5:LU IS NOT ACTIVE	An LU is not active. See the LUNAME field in the trace output. Enter a DISPLAY APPC command to find the status of this LU.
FMH5NOTP	FMH5:TP NAME IS NOT RECOGNIZED	The TP name was not specified correctly in the FMH-5.
FMH5NSCH	FMH5:NOT SERVED AND NO SCHEDULER	The TP cannot be scheduled because no scheduler is associated with the LU.
FMH5PFST	FMH5:FMH-5 PROFILE IS NOT VALID	The FMH-5 profile is incorrect; it is greater than 8 characters.
FMH5PIP	FMH5:PIP DATA PRESENT IN FMH-5	APPC/MVS found profile initialization parameters (PIP) data in the FMH-5; PIP data is not valid in FMH-5 for APPC/MVS.
FMH5PWCC	FMH5:PW CONV CLEANUP FAILED	Internal error. Contact the IBM Support Center.
FMH5PWDE	FMH5:PW DEALLOCATE FAILED	Internal error. Contact the IBM Support Center.
FMH5PWDF	FMH5:PW DEQUE REQUEST FAILED	An attempt to attach the SIGNON/Change password TP failed. Contact the IBM Support Center.
FMH5PWQF	FMH5:PW QUEUE REQUEST FAILED	Internal error. Contact the IBM Support Center.
FMH5PWRF	FMH5:QW RACF REQUEST REJECTED	Internal error. Contact the IBM Support Center.
FMH5PWR1 FMH5PWR2	FMH5:PW RECEIVE DATA FAILED 1 FMH5:PW RECEIVE DATA FAILED 2	__The SIGNON/Change password TP attempted to perform a ReceiveandWait call for a GDS variable. See the following KEY fields: • **KEY X'007E'** contains the status received__ value returned to the SIGNON/Change password TP by ReceiveandWait. • **KEY X'007F'** contains the data received value__ returned to the SIGNON/Change password TP by ReceiveandWait. • **KEY X'003F'** contains the return code from__ ReceiveandWait. Make sure that your GDS variable was sent correctly. If you cannot resolve the problem, contact the IBM Support Center.
FMH5PWSD	FMH5:PW SEND DATA FAILED	_A SIGNON/Change password TP SendData call failed. Verify that your TP has a valid conversation established with the SIGNON/Change password TP. If you cannot resolve the problem, contact the IBM Support Center.

Table 11-1. FMH-5 Trace Entries in the SYSAPPC Component Trace (continued)

Mnemonic	Title	Description/Action
FMH5PWSF	FMH5:PW SEND MESSAGE FAILED	A persistent verification signoff flow to the partner LU failed. The__ SIGNEDONTO list in the partner LU may not be in sync with the local__ SIGNEDONFROM list. See the following KEY fields: • **KEY X'0026'** contains the TCB address. • **KEY X'001A'** contains the name of the partner LU. • **Key X'002F'** contains the userid of the user whose SIGNOFF failed. If you cannot resolve the problem, contact the IBM Support Center.
FMH5PWSM	FMH5:PW SEND MESSAGE	APPC/MVS could not attach the X'30F0F5F2' expired password notification program to notify a partner system user that the user’s password expired. See the following KEY fields: • **KEY X'0026'** contains the TCB address. • **KEY X'001A'** contains the name of the partner LU. • **KEY X'002F'** contains the USERID of the user whose attach request failed. If you cannot resolve the problem, contact the IBM Support Center.
FMH5PWSR	FMH5:PW SIF RESERVE FAILURE	Internal error. Contact the IBM Support Center.
FMH5PWST	FMH5:FMH-5 PASSWORD IS NOT VALID	The FMH-5 password is incorrect; it is greater than 8 characters.
FMH5QMFL	FMH5:FMFP QUEUE MANAGER FAILURE	Internal error. Contact the IBM Support Center
FMH5RECV	FMH5:FMH-5 SUCCESSFULLY RECEIVED	An FMH-5 was successfully received by the local MVS LU.

Table 11-1. FMH-5 Trace Entries in the SYSAPPC Component Trace (continued)

Mnemonic	Title	Description/Action
FMH5RFRJ	FMH5:RACF REQUEST REJECTED	The system received a bad return code from one of the RACF services. See KEY X'0053' for a code identifying the RACF service that failed. The code can be one of the following: **1** RACROUTE REQUEST=VERIFY **2** RACROUTE REQUEST=SIGNON TYPE=SIGNIN **3** RACROUTE REQUEST=SIGNON TYPE=QSIGNON **4** RACROUTE REQUEST=SIGNON TYPE=SIGNOFF See the following KEY fields: • **KEY X'0054'** contains the return code for the RACF service request. • **KEY X'0055'** contains the reason code for the RACF service request. • **KEY X'0021'** contains the security authorization facility (SAF) return code for the service.
FMH5SERF	FMH5:APPC/MVS SERVICE FAILURE	APPC/MVS internal failure. Contact the IBM Support Center.
FMH5SFAL	FMH5:SEND MESSAGE FAILED	Persistent verification signoff flow to the partner LU failed. Make sure you have valid sessions established. See the following KEY fields: • **KEY X'0026'** contains the TCB address. • **KEY X'001A'** contains the name of the partner LU.
FMH5SOFF	FMH5:SIGNOFF FLOW	Persistent verification signoff flow to the partner LU completed. See the following KEY fields: • **KEY X'0026'** contains the TCB address. • **KEY X'001A'** contains the name of the partner LU.
FMH5SVFC	FMH5:ACCEPTED BY SRVR FACILITIES	APPC/MVS placed the inbound request on an allocate queue to await later processing by an APPC/MVS server.
FMH5TEST	FMH5:FMH5 ACCEPTED FOR TESTING	An FMH-5 is accepted for testing.
FMH5TPAD	FMH5:TP PROFILE ACCESS DENIED	TP profile access denied. Request=AUTH failed.
FMH5TPNA	FMH5:TP PROFILE IS NOT ACTIVE	The TP profile is not active. Get the TP name from the FMH-5 formatted at KEY X'0012' in this trace record. Then use the SDFM utility to look at the TP profile.
FMH5TPRQ	FMH5:TP PROFILE IS REQUIRED	The system found no TP profile for the requested TP. The scheduler associated with the TP requires a TP profile. The error is probably due to an SDFM problem. Look for trace records with a prefix of SDFM.

Table 11-1. FMH-5 Trace Entries in the SYSAPPC Component Trace (continued)

Mnemonic	Title	Description/Action
FMH5UIST	FMH5:FMH-5 USERID IS NOT VALID	The FMH-5 userid is incorrect; it is greater than 8 characters.
FMH5VALD	FMH5:FMH5 SUCCESSFULLY VALIDATED	An FMH-5 has been successfully validated.
FMH5XLNF	FMH5:EXCHANGE LOG NAME FAILED	APPC/MVS rejected the protected conversation because required log-name exchange processing did not occur.
QMANFAIL	FMH5:FMAX QUEUE MANAGER FAILURE	Internal error. Contact the IBM Support Center.
RESVFAIL	FMH5:SIF RESERVE FAILURE	Internal error. Contact the IBM Support Center.

SYSDLF Component Trace

Before Using This Component Trace

This topic assumes you have read:

- "Planning for Component Tracing" on page 11-3
- "Obtaining a Component Trace" on page 11-10
- "Viewing the Component Trace Data" on page 11-23

The following summarizes information for requesting a SYSDLF component trace for the data lookaside facility (DLF).

Information	For SYSDLF:
Parmlib member	None
Default tracing	Yes; always on when DLF is running
Trace request OPTIONS parameter	None
Buffer	• Default: N/A • Range: N/A • Size set by: MVS system • Change size after IPL: No • Location: Data space. In the REPLY for the DUMP command, specify DSPNAME=('DLF'.CCOFGSDO)
Trace records location	Address-space buffer, data-space buffer
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSDLF)
Trace format OPTIONS parameter	None

Requesting a SYSDLF Trace

The trace runs whenever DLF is in control. No actions are needed to request it.

Formatting a SYSDLF Trace

Format the trace with an IPCS CTRACE COMP(SYSDLF) subcommand. The subcommand has no OPTIONS values.

Output from a SYSDLF Trace

CTRACE COMP(SYSDLF) FULL Subcommand Output

The following is an example of DLF component trace records formatted with a CTRACE COMP(SYSDLF) FULL subcommand. It shows formatted exception records from the trace buffers.

```
                    DLF COMPONENT TRACE FULL FORMAT

 COFRCVRY  00000000  15:47:40.397545  DLF RECOVERY ENTRY
   HASID... 000E     SASID... 000E     CPUID... FF170067 30900000
   MODNAME. COFMCON2 ABEND... 840C1000 REASON.. 00000001
   EPTABLE. CON2  EST2  ....  ....  ....  ....  ....  ....  ....  ....
 COFRCVRY  00000001  15:47:40.397625  DLF RECOVERY EXIT
   HASID... 000E     SASID... 000E     CPUID... FF170067 30900000
   MODNAME. COFMCON2 ABEND... 840C1000 REASON.. 00000001
   RETCODE. 0000002C RSNCODE. 0000C200 FTPRTS.. C0000000 DATA.... 00000000
```

The fields in the report are:

COFRCVRY
: The name or identifier of the trace record.

00000000
: The identifier in hexadecimal.

15:47:40.397545
: The time stamp indicating when the record was placed in the trace table.

HASID... 000E
: The home address space identifier.

SASID... 000E
: The secondary address space identifier.

CPUID... FF170067 30900000
: The identifier of the processor that placed the record in the trace table.

CALLER
: The address of the routine that issued a DLF service request.

MODNAME COFMCON2
: The name of the module that was running.

ABEND... 840C1000
: The abend that occurred and caused DLF to enter recovery.

REASON.. 00000001
: The reason code associated with the abend.

EPTABLE. CON2 EST2
: Information used for diagnosis by IBM.

RETCODE. 0000002C
: The return code that was issued by the module that is exiting.

RSNCODE. 0000C200
: The reason code that was issued by the module that is exiting.

FTPRTS.. C0000000
: Information used for diagnosis by IBM.

DATA.... 00000000
: Information used for diagnosis by IBM.

SYSDSOM Component Trace

Before Using This Component Trace

This topic assumes you have read:
- "Planning for Component Tracing" on page 11-3
- "Obtaining a Component Trace" on page 11-10
- "Viewing the Component Trace Data" on page 11-23

The following summarizes information for requesting a SYSDSOM component trace for distributed SOMobjects (DSOM).

Information	For SYSDSOM:
Parmlib member	None
Default tracing	No
Trace request OPTIONS parameter	In REPLY for TRACE command
Buffer	• Default: N/A • Range: N/A • Size set by: MVS system • Change size after IPL: No • Location: Address space
Trace records location	Address-space buffer
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSDSOM)
Trace format OPTIONS parameter	Yes

Requesting a SYSDSOM Trace

Request the trace by specifying any non-zero value on the SOMDTRACELEVEL DSOM environment variable.

Formatting a SYSDSOM Trace

Format the trace with an IPCS CTRACE COMP(SYSDSOM) subcommand. The subcommand has the following OPTIONS values:

SKIPID
Omits the jobname, ASID, and thread identifier from the output.

LONGFORM
Tells the system to display detailed output. In the output, each trace function might have multiple trace elements, each on a separate line. If you do not specify LONGFORM, the default is SHORTFORM. Do not specify both LONGFORM and SHORTFORM on the OPTIONS parameter.

SHORTFORM
Tells the system to display abbreviated output. In the output, each trace function is on one line. SHORTFORM is the default value. Do not specify both LONGFORM and SHORTFORM on the OPTIONS parameter.

Output from a SYSDSOM Trace

CTRACE COMP(SYSDSOM) FULL Subcommand Output

The following is an example of DSOM component trace records formatted with a CTRACE COMP(SYSDSOM) FULL OPTIONS((SKIPID,SHORTFORM)) subcommand. It shows formatted exception records from the trace buffers.

```
                    DSOM COMPONENT TRACE FULL FORMAT

     KESYS522  METRETRN  00000004  21:31:34.864277  Return from method
          Entry to method:        ImplRepository::somInit
```

The following is an example of DSOM component trace records formatted with a CTRACE COMP(SYSDSOM) FULL subcommand. It shows formatted exception records from the trace buffers.

```
COMPONENT TRACE FULL FORMAT
SYSNAME(KESYS522)
COMP(SYSDSOM)
**** 09/29/1995
SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------
KESYS522  GETBUFF   00000001  21:31:30.198289  Get new trace buffer
     JOBNAME. KREPROC  ASID.... 0029     THREADID 04233100 00000000
     Buffer address:         7F672508
KESYS522  METDEBUG  00000004  21:31:39.410681  Method debug
     JOBNAME. KREPROC  ASID.... 0029     THREADID 04233100 00000000
     Entry to method:        SOMOA::somInit
KESYS522  METRETRN  00000005  21:31:40.000019  Return from method
     JOBNAME. KREPROC  ASID.... 0029     THREADID 04233100 00000000
     Exiting method:         SOMOA::somInit, RC(hex)=00000000, RSN=00000000
```

The following is an example of DSOM component trace records formatted with a CTRACE COMP(SYSDSOM) FULL OPTIONS((SKIPID)) DSN('dsom.trace.dsn') subcommand. It shows formatted exception records from the trace buffers.

```
COMPONENT TRACE FULL FORMAT
SYSNAME(KESYS522)
COMP(SYSDSOM)
OPTIONS((SKIPID))
**** 09/29/1995
SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------
KESYS522  GETBUFF   00000001  21:31:30.198289  Get new trace buffer
     Buffer address:         7F672508
KESYS522  METDEBUG  00000004  21:31:39.410681  Method debug
     Entry to method:        SOMOA::somInit
KESYS522  METRETRN  00000005  21:31:40.000019  Return from method
     Exiting method:         SOMOA::somInit, RC(hex)=00000000, RSN=00000000
```

The fields in the report are:

KESYS522
The name of the system.

METDEBUG
The name of the trace event.

00000004
The decimal identifier of the trace event.

21:31:39.410681
The time stamp indicating when the record was placed in the trace table.

ASID
The ASID of the job listed in the JOBNAME field.

JOBNAME. KREPROC
The job name.

THREADID 04233100 00000000
The POSIX thread identifier.

Entry to method
The entry to the somInit method in class SOMOA.

Exiting Method
The exit from the somInit method in class SOMOA.

SYSGRS Component Trace

Before Using This Component Trace

This topic assumes you have read:
- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSGRS component trace for global resource serialization.

Information	For SYSGRS:
Parmlib member	CTnGRSxx Default member: CTIGRS00 specified in GRSCNF00 member
Default tracing	Yes, if global resource serialization **ring** is active; CONTROL and MONITOR options
Default tracing	Yes, if global resource serialization **star** is active; CONTROL1, CONTROL2, SIGNAL0 and MONITOR options
Trace request OPTIONS parameter	In CTnGRSxx and REPLY for TRACE command
Buffer	• Default: 128KB • Range: 128KB - 16MB (System rounds size up to nearest 64KB boundary.) • Size set by: CTnGRSxx member • Change size after IPL: Yes, when restarting a trace after stopping it • Location: In the component address space
Trace records location	Address-space buffer, trace data set
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSGRS)
Trace format OPTIONS parameter	FLOW, CONTROL, MONITOR, REQUEST, SIGNAL, and RSA

Requesting a SYSGRS Trace

Specify options for requesting a SYSGRS component trace on a CTnGRSxx parmlib member or on the reply for a TRACE CT command.

You can change options for SYSGRS tracing while the trace is running.

CTnGRSxx Parmlib Member

The following table indicates the parameters you can specify on a CTnGRSxx parmlib member.

Parameters	Allowed on CTnGRSxx?
ON or OFF	Yes
ASID	No
JOBNAME	No
BUFSIZE	Yes
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

IBM supplies the CTIGRS00 parmlib member, which specifies the GRS tracing begun at IPL. The contents of CTIGRS00 are:

```
TRACEOPTS
     OFF
```

This parameter turns off all SYSGRS tracing options except for the minimum options (MINOPS).

If additional SYSGRS tracing options are turned on, additional buffer space may be required. If any FLOW or MONITOR options are used, buffer size of at least 16 megabytes is recommended. These options request the unexpected or important global resource serialization events.

The default trace buffer size is 128KB. In the IBM-supplied GRSCNF00 parmlib member, the CTRACE parameter specifies CTIGRS00 as the default.

IBM recommends that you use the CTIGRS00 parmlib member, unless the IBM Support Center requests different tracing for global resource serialization.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON, nnnnK, nnnnM, or OFF	One is required
COMP	Required
SUB	No
PARM	Yes

Parameters	Allowed on TRACE CT for Writer?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	No
JOBNAME	No
OPTIONS	Yes
WTR	Yes

You can change options while a SYSGRS trace is running. However, to change the buffer size, you have to stop the trace and restart it with the new buffer size.

OPTIONS Parameter

The values for the OPTIONS parameter for the CTnGRSxx parmlib member and reply for a TRACE command are listed below. The sub-options on the CONTROL, REQUEST, MONITOR, SIGNAL and FLOW, allow you to refine the set of events traced for the major option. When you select the major option, all events pertaining to that option are traced. However, you can select one or more of the sub-options instead of the major option and thus limit the trace to only those events included in the sub-options specified. A major option, such as MONITOR, and all of its sub-options (in this case MONITOR0, MONITOR1, and MONITOR2 through MONITORF) is referred to as an option group. In alphabetical order the values for the OPTIONS parameter are:

CONTROL

Traces unusual events and events related to the establishment, modification, or termination of the control structure needed for processing such as:

- Dynamic RNL changes
- Error events
- XCF services used when setting up for processing

When you specify CONTROL, all of the following sub-options are traced.

CONTROL0

Traces dynamic RNL changes only.

CONTROL1

Traces events related to the establishment of or termination of membership in the global resource serialization group connection to the global resource serialization coupling facility structures.

CONTROL2

Traces global resource serialization recovery processing only.

CONTROL3

Traces global resource serialization resource manager events for abnormal task and ASID termination only.

CONTROL4-CONTROLE

Reserved for IBM use.

CONTROLF

Traces all other unusual events not included in sub-options CONTROL0 through CONTROL3.

FLOW

Traces the flow of control from one entry point to another.

FLOW0
Traces GRS Star system server processing only.

FLOW1
Traces GQSCAN processing only.

FLOW2
Traces cross-system communications processing only.

FLOW3
Traces command processing only.

FLOW4
Traces storage manager services only.

FLOW5
Traces coupling facility processing only.

FLOW6
Traces initialization processing only.

FLOW7
Traces contention monitor processing only.

FLOW8
Traces general ENQ/DEQ processing only.

FLOW9-FLOWD
Reserved for IBM use.

FLOWE
Activates extended tracing for the GRS Storage Manager. Do not turn on this option without direction from IBM Service.

FLOWF
Reserved for IBM use.

Monitor
Traces events for selected global resource serialization invocations of monitoring and communication services provided by other components.

MONITOR0
Traces use of XES services.

MONITOR1
Traces use of XCF services.

MONITOR2-MONITORF
Reserved for IBM use.

REQUEST
Traces events for global ENQ, DEQ, GQSCAN, and RESERVE macro requests, and GRS command processing.

REQUEST0
Traces ENQ/RESERVE requests only.

REQUEST1
Traces DEQ requests only.

REQUEST2
Traces GQSCAN only.

REQUEST3
Traces IXLLOCK only.

REQUEST4
Traces command processing only.

REQUEST5
Traces lock structure (ISGLOCK) rebuild processing only.

REQUEST6-REQUESTF
Reserved for IBM use.

RSA
Traces events for RSA control information.

SIGNAL
Traces events for selected global resource serialization invocations of cross-system coupling facility (XCF) signalling service processing.

SIGNAL0
Traces migration signals only.

SIGNAL1
Traces GQSCAN signals only.

SIGNAL2
Traces ENQ/DEQ signals, including RNL change signals only.

SIGNAL3
Traces contention monitor signals only.

SIGNAL4-SIGNALF
Reserved for IBM use.

Examples of Requesting SYSGRS Traces

Example 1: CTnGRSxx Member

The member requests CONTROL, MONITOR, and RSA options and doubles the default buffer size.

```
TRACEOPTS
   ON
   OPTIONS('CONTROL','MONITOR','RSA')
   BUFSIZE(256K)
```

Example 2: TRACE Command

The example requests a trace of CONTROL, MONITOR, and REQUEST trace events.

```
trace ct,on,comp=sysgrs
* 17 ITT006A ...
reply 17,options=(control,monitor,request),end
```

Formatting a SYSGRS Trace

Format the trace with an IPCS CTRACE COMP(SYSGRS) subcommand. It is possible to use the OPTIONS subcommand for COMP (SYSGRS) with values of FLOW, CONTROL, REQUEST, MONITOR, SIGNAL, and RSA for filtering.

Output from a SYSGRS Trace

CTRACE COMP(SYSGRS) SHORT Subcommand Output

The following is an example of SYSGRS component trace records formatted with the CTRACE COMP(SYSGRS) SHORT subcommand.

```
                   SYSGRS COMPONENT TRACE SHORT FORMAT

    GRPXCNTL  00000030  13:05:59.858746  GROUP EXIT IN CONTROL
    DISRUPT   0000000E  13:05:59.858780  RING DISRUPTION TRIGGERED
    MAINRF1   0000000B  13:05:59.860196  MAIN RING FAILURE
    CEXBCI1   0000000C  13:05:59.860324  CONTROL EXITED FROM ISGBCI
    SETUS     00000035  13:06:00.031243  CALL TO XCF SETUS SERVICE
    GRPXCNTL  00000030  13:06:00.141669  GROUP EXIT IN CONTROL
    STAXIN    00000033  13:06:00.160559  STATUS EXIT IN CONTROL
    .
    .
    .
```

CTRACE COMP(SYSGRS) TALLY Subcommand Output

The following is an example of SYSGRS component trace records formatted with the CTRACE COMP(SYSGRS) TALLY subcommand.

```
  COMPONENT TRACE TALLY REPORT
  COMP(SYSGRS)
      TRACE ENTRY COUNTS AND AVERAGE INTERVALS (IN MICROSECONDS)

  FMTID     COUNT     INTERVAL      MNEMONIC  DESCRIBE
  --------  --------  ------------  --------  --------------------------------
  00000001         0                RSAIN1    RSA has no CMD area no QWB data
  00000002         0                RSAIN2    RSA has QWB data but no CMD area
  00000003         0                RSAIN3    RSA has CMD area but no QWB data
  00000004         0                RSAIN4    RSA has CMD area and QWB data
  00000005         0                RSAOUT1   RSA has no CMD area no QWB data
  00000006       898     2,037,854  RSAOUT2   RSA has QWB data but no CMD area
  00000007         0                RSAOUT3   RSA has CMD area but no QWB data
  00000008         8   149,924,356  RSAOUT4   RSA has CMD area and QWB data
  00000009         5   328,792,726  INVBBE1   ISGBBE - QMERGE
  0000000A         0                INVBBE2   ISGBBE - not QMERGE
  0000000B         4   490,847,959  MAINRF1   Main Ring Failure
  0000000C        13   149,346,135  CEXBCI1   Control Exited from ISGBCI
  0000000D         0                CLNQSCD   Cleanup after Quiesced from ring
  .                                 .         .
  .                                 .         .
  .                                 .         .

  00000058         0                UEVENT8   Recovery during write
  00000059         0                UEVENT9   Recovery during read
  0000005A         0                UEVENTA   Remote discarded message

 Total trace entries:    1,120
```

SYSIEFAL Component Trace

Before Using This Component Trace

This topic assumes you have read:
- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSIEFAL component trace for Allocation.

Information	For SYSIEFAL:
Parmlib member	CTIIEFxx Default member: CTIIEFAL
Default tracing	Yes; FLOW0, FLOW1, DATA, CONTROL0, CONTROL1, and SERIAL1 options
Trace request OPTIONS parameter	In CTIIEFxx or REPLY for TRACE command
Buffer	• Default: 4M • Range: 16KB - 8MB • Size set by: CTIIEFxx member • Change size after IPL: Yes • Location: In the component address space.
Trace records location	Address-space buffer
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSIEFAL)
Trace format OPTIONS parameter	Yes; FLOW, CONTROL, SERIAL, and DATA

Requesting a SYSIEFAL Trace

Specify options for requesting a SYSIEFAL component trace on a CTIIEFxx parmlib member or on the reply for a TRACE CT command.

You can change options for SYSIEFAL tracing while the trace is running.

CTIIEFxx Parmlib Member

The following table indicates the parameters you can specify on a CTIIEFxx parmlib member.

Parameters	Allowed on CTIIEFxx?
ON or OFF	Yes
ASID	Yes
JOBNAME	Yes
BUFSIZE	Yes
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No

Parameters	Allowed on CTIIEFxx?
WTR	Yes
WTRSTART or WTRSTOP	Yes

IBM supplies the CTIIEFAL parmlib member, which specifies the Allocation tracing begun at IPL. The contents of CTIIEFAL are:

```
TRACEOPTS
       ON
OPTIONS(
        'FLOW0'
       ,'FLOW1'
       ,'SERIAL1'
       ,'DATA'
       ,'CONTROL0'
       ,'CONTROL1'
       )
BUFSIZE(4M)
```

If additional SYSIEFAL tracing options are turned on, additional buffer space may be required. If any FLOW, SERIAL, or CONTROL options are used, a buffer size of 8 megabytes is recommended. These options request the unexpected or important allocation events.

The default trace buffer size is 4M.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON, nnnnK, nnnnM, or OFF	One is required
COMP	Required
SUB	No
PARM	Yes

Parameters	Allowed on TRACE CT for Writer?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	Yes
JOBNAME	Yes
OPTIONS	Yes
WTR	Yes

You can change options while a SYSIEFAL trace is running. However, to change the buffer size, you have to stop the trace and restart it with the new buffer size.

OPTIONS Parameter

The values for the OPTIONS parameter for the CTIIEFxx parmlib member and reply for a TRACE command are listed below. The sub-options on the CONTROL, DATA, FLOW, and SERIAL, allow you to refine the set of events traced for the major option. When you select the major option, all events pertaining to that option are traced. However, you can select one or more of the sub-options instead of the major option and thus limit the trace to only those events included in the sub-options specified. A major option, such as DATA, and all of its sub-options (in this case DATA0, DATA1, and so forth) is referred to as an option group. In alphabetical order the values for the OPTIONS parameter are:

CONTROL
Traces the control within a module.

CONTROL0
Traces Common Allocation processing only.

CONTROL1
Traces Allocation Services 1 processing only.

CONTROL2
Traces unallocation processing only.

CONTROL3
Traces Volume Mount and Verify processing only.

CONTROL4
Traces Assign/Unassign processing only.

CONTROL5
Traces Allocation Services 2 processing only.

CONTROL6–CONTROLE
Reserved for IBM use.

CONTROLF
Traces unexpected or unusual control within a module.

DATA
Traces when data is processed or changed.

DATA0
Traces when an ATSP Device Type Array is being processed only.

DATA1
Traces when an ATSP Device Array is being processed only.

DATA2
Traces when an IGDE is going through XCF messaging only.

DATA3
Traces when an IGDE goes through a state change only.

DATA4–DATAE
Reserved for IBM use.

DATAF
Traces when data is unexpected or unusual.

FLOW
Traces the flow of control from one entry point to another.

FLOW0
Traces Common Allocation processing only.

FLOW1
Traces Allocation Services 1 processing only.

FLOW2
Traces unallocation processing only.

FLOW3
Traces Volume Mount and Verify processing only.

FLOW4
Traces Assign/Unassign processing only.

FLOW5
Traces Allocation Services 2 processing only.

FLOW6–FLOWE
Reserved for IBM use.

FLOWF
Traces unexpected or unusual flow from one entry point to another.

SERIAL
Traces serialization events.

SERIAL0
Traces locking (SETLOCK) serialization events only.

SERIAL1
Traces ENQ/DEQ serialization events only.

SERIAL2
Traces Latch Manager serialization events only.

SERIAL3
Traces Compare and Swap serialization events only.

SERIAL4–SERIALE
Reserved for IBM use.

SERIALF
Traces unexpected or unusual serialization events.

Examples of Requesting SYSIEFAL Traces

Example 1: CTIIEFxx Member

The member requests FLOW and DATA options and requests a buffer size of 8 megabytes.

```
TRACEOPTS
   ON
   OPTIONS('FLOW','DATA')
   BUFSIZE(8M)
```

Example 2: TRACE Command

The example requests a trace of DATA1 and CONTROL1 trace events.

```
trace ct,on,comp=sysiefal
* 17 ITT006A ...
reply 17,options=(data1,control1),end
```

Formatting a SYSIEFAL Trace

Format the trace with an IPCS CTRACE COMP(SYSIEFAL) subcommand. It is possible to use the OPTIONS subcommand for COMP (SYSIEFAL) with values of FLOW, DATA, and SERIAL for filtering.

Output from a SYSIEFAL Trace

CTRACE COMP(SYSIEFAL) SHORT Subcommand Output

The following is an example of SYSIEFAL component trace records formatted with the CTRACE COMP(SYSIEFAL) SHORT subcommand.

```
COMPONENT TRACE SHORT FORMAT
COMP(SYSIEFAL)
**** 09/13/2001

SYSNAME MNEMONIC ENTRY ID   TIME STAMP    DESCRIPTION
------- -------- -------- --------------- -------------
N67     FLOW0    00000100 15:40:34.104633 Common Allocation Flow
N67     CONTROL0 00000200 15:40:34.104639 Common Allocation Control
N67     CONTROL0 00000200 15:40:34.104693 Common Allocation Control
N67     CONTROL0 00000200 15:40:34.104852 Common Allocation Control
N67     FLOW0    00000100 15:40:34.104859 Common Allocation Flow
N67     FLOW0    00000100 15:40:34.106631 Common Allocation Flow
N67     FLOW0    00000100 15:40:34.125582 Common Allocation Flow
.
.
.
```

CTRACE COMP(SYSIEFAL) FULL Subcommand Output

The following is an example of SYSIEFAL component trace records formatted with the CTRACE COMP(SYSIEFAL) FULL subcommand.

```
COMPONENT TRACE FULL FORMAT
COMP(SYSIEFAL)
**** 09/13/2001

SYSNAME MNEMONIC ENTRY ID   TIME STAMP    DESCRIPTION
------- -------- -------- --------------- -------------
N67     FLOW0    00000100 15:40:47.306228 Common Allocation Flow

   ASID..007A  TCB..008E1980  MODNAME..IEFAB492  JOBNAME..T015067

    EBCDIC Data...
    ENTR

N67     FLOW0    00000100 15:40:47.306231 Common Allocation Flow

   ASID..007A  TCB..008E1980  MODNAME..IEFAB492  JOBNAME..T015067

    EBCDIC Data...
    EXIT
    Hex Data...
    00000000                                | ....              |

.
.
.
```

SYSIOS Component Trace

Before Using This Component Trace

This topic assumes you have read:

- "Planning for Component Tracing" on page 11-3
- "Obtaining a Component Trace" on page 11-10
- "Viewing the Component Trace Data" on page 11-23

IOS component trace is described by the following attributes:

- Trace buffers reside in common ESQA Subpool 248. Size is controlled by the TRACE CT operator command. As the buffers become full they are copied to a private IOS data space.
- Minimal and unexpected event tracing is activated during IOS NIP processing.
- Component trace buffers externalized via:
  - DUMP or SLIP operator command when the IOS address space is requested to be dumped.
  - SVC dumps issued by IOS, dynamic device reconfiguration (DDR), or execute channel program (EXCP) component recovery.
  - MVS component trace (CTRACE) external writer
- Trace options revert to minimal event and exception tracing when the operator turns the trace off.

The following summarizes information for requesting a SYSIOS component trace for IOS:

Information	**For SYSIOS:**
Parmlib member	CTnIOSxx specified in the IECIOSxx member via the CTRACE(CTnIOSxx) statement. Default member: None
Default tracing	Yes, activated during IOS NIP processing.
Trace request OPTIONS parameter	In CTnIOSxx or REPLY for TRACE command
Buffer	• Default: 36KB • Range: 36KB - 1.5M • Size set by: CTnIOSxx member or REPLY for TRACE command • Change size after IPL: Yes, when component trace (CTRACE) is active • Location: Common ESQA subpool 248 and SYSIOS private IOS data space.
Trace records location	Common ESQA subpool 248 and SYSIOS private IOS data space and trace data set. By DUMP or SLIP command when the IOS address space is requested to be dumped. In the REPLY for the DUMP command, specify the IOS address space to be dumped. By SLIP command. By the component during SVC dumps issued by IOS, DDR, or EXCP component recovery.
Trace formatting by IPCS	CTRACE COMP(SYSIOS)

Information	For SYSIOS:
Trace format OPTIONS parameter	No

The areas of IOS traced as part of minimal and unexpected event tracing include:
- Dynamic Configuration Changes
- Parallel Access Volume (PAV) Processing
- Dynamic Channel Path Management (DCM) Processing
- Unconditional Reserve (U/R) Recovery Processing
- Channel Subsystem Call (CHSC) Processing
- Channel Report Word (CRW) Processing
- Missing Interrupt Handler (MIH) Recovery Processing
- Control Unit Initiated Reconfiguration (C.U.I.R.) Request Processing
- Dynamic Pathing Support (DPS) Validation
- Dynamic Device Reconfiguration (DDR) Processing
- Self-Description Processing

Note: Additional areas are traced when OPTIONS are set for IOS component trace. See "OPTIONS Parameter" on page 11-56.

Requesting a SYSIOS Trace

No actions are required to request a SYSIOS trace. Minimal and unexpected event tracing is activated during IOS NIP processing and is always active. In addition to this tracing, the user can request a SYSIOS trace using specific options by doing the following:
- Using a CTnIOSxx SYS1.PARMLIB member during NIP processing by specifying the CTRACE(CTnIOSxx) statement in the IECIOSxx SYS1.PARMLIB member.
- Using a CTnIOSxx SYS1.PARMLIB member after NIP by issuing the TRACE system command.
- Using the TRACE system command and specifying the options in response to the prompts that CTRACE provides.

CTnIOSxx Parmlib Member

The following table indicates the parameters you can specify in a CTnIOSxx parmlib member.

Parameters	Allowed on CTnIOSxx?
ON or OFF	One is required
ASID	Yes
JOBNAME	Yes
BUFSIZE	Yes
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

If additional SYSIOS tracing options are turned on, additional buffer space might be required.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON or OFF	One is required
nnnnK, nnnnM	Yes
COMP	Required
SUB	No
PARM	Yes

Parameters	Allowed on TRACE CT for Writer?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	Yes
JOBNAME	Yes
OPTIONS	Yes
WTR	Yes

The WTR and WTRSTART parameters can be used in a parmlib member specified on the TRACE CT command. The parameters cannot be specified in a parmlib member that is read at IPL because the external writer is not available when the IOS component is defined.

OPTIONS Parameter

The values for the OPTIONS parameter for the CTnIOSxx parmlib member and reply for a TRACE command are listed below.

DCM
Traces events relating to Dynamic Channel Path Management.

Note: When DCM is active and this option is set, large amounts of trace data will be recorded. Users may wish to consider using an external writer when this option is set.

Traces the results of the cancel subchannel (XSCH) instruction.

EXTEND
Traces all functions that are attached in the IOS address space.

Note: Some functions attached in the IOS address space are traced during minimum or unexpected event tracing.

Traces the results of the cancel subchannel (XSCH) instruction.

STORAGE
Traces events related to IOS or EXCP storage management. When the STORAGE option is specified, the NOFILTER option or ASID/JOBNAME

keywords must also be set. When the STORAGE option is in effect, CTRACE will automatically increase the buffer size to 252K.

NOFILTER
Allows the STORAGE option to be set without requiring ASID or JOBNAME filtering.

DS=nnnn
Allows the user to tailor the IOS Trace Data Space where nnnn is the data space size in megabytes.

Notes:

1. nnnn must be a valid decimal digit within the range of 1-1024.
2. This option can only be specified once at IPL time and cannot be modified using the TRACE CT command.
3. The default size for the IOS Trace Data Space is 512M. This can require additional auxiliary storage on systems with a small amount of available auxiliary storage. Please refer to "Decide Where to Collect the Trace Records" on page 11-9 for information about auxiliary storage for CTRACE data space buffers. Since some options such as STORAGE and DCM will cause more CTRACE entries to be recorded, the IOS Trace Data Set may fill up more rapidly than if none of these options is specified. Users who do not have enough auxiliary storage capacity to handle a full data space may choose to use the DS=nnnn option to set up a smaller IOS Trace Data Space. Similarly, users who do have enough auxiliary storage capacity to handle a full data space may choose to use the DS=nnnn option to set up a larger IOS Trace Data Space. Doing this will prevent potentially valuable debug information from being lost due to wrapping. Note that the number of records contained in a trace data set are highly variable and dependent upon trace settings and system usage.
4. If the DS=nnnn option is specified more than once, the request is rejected and the following message is issued:

```
IOS622I IOS COMPONENT TRACE OPTION xxxxxxxx IS NOT VALID -
        THE TRACE DATA SPACE SIZE HAS ALREADY BEEN SET
        FOR THIS IPL
```

5. If the size specified is not valid, then the request is rejected, the default IOS Trace Data Space size is used, and the following message is issued:

```
IOS622I IOS COMPONENT TRACE OPTION xxxxxxxx IS NOT VALID -
        THE REQUESTED SIZE FOR THE TRACE DATA SPACE IS
        INCORRECT
```

Examples of Requesting SYSIOS Traces

Example 1: CTnIOSxx SYS1.PARMLIB Member

This SYS1.PARMLIB member sets the STORAGE option using JOBNAME filtering for JOB001 and sets the buffer size to 600K.

```
TRACEOPTS
   ON
   BUFSIZE(600K)
   OPTIONS(STORAGE)
   JOBNAME(JOB001)
```

Formatting a SYSIOS Trace

IPCS CTRACE formatting services can be used to format the contents of the CTRACE trace entries. Format the trace with the following IPCS subcommand:

```
{|||}IPCS CTRACE COMP(SYSIOS) SUMMARYFULLSHORTTALLY
```

Where:

SUMMARY Shows the trace entry header and the formatted data for each trace entry.

FULL Shows the trace entry header and the unformatted (hex) data for each trace entry.

SHORT Shows the trace entry header for each trace entry.

TALLY Shows each trace entry and how many times they were traced.

The subcommand has no options.

CTRACE COMP(SYSIOS) Subcommand Output

The following is an example of SYSIOS component trace records formatted with the CTRACE COMP(SYSIOS) SUMMARY option.

```
CTRACE COMP(SYSIOS) SUMMARY
**** 03/28/1996
SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------

S530      MIH       00080001  19:42:42.220726  MIH Recovery Halt/Clear Block

 Trace Record Function: MIH
  MIH condition detected: Start Pending

 Record ID: IOSDMHCB.MHCBSHIB              Length: 0034

   +0000  00F168E8  289B0180  F00000F0  0027FFF0  | .1.Y....0..0...0 |
   +0010  3868B8E8  FFFFFFFF  00000000  00804400  | ...Y............ |
   +0020  3885DA88  00000000  00000000  | .e.h............ |
   +0030  00000000                                | ....             |

 Record ID: IOSDMHCB.MHCBUCB               Length: 0080

   +0000  00000940  20200000  01E13480  00000000  | ... ............ |
   +0010  00000000  00FCC8B4  00F16890  00000000  | ......H..1...... |
   +0020  00040040  00FC3800  00FC3100  0001001F  | ... ............ |
   +0030  28980027  F00080F0  3868B8E8  FFFFFFFF  | .q..0..0...Y.... |
   +0040  01000040  00000001  00000041  00FC3800  | ... ............ |
   +0050  0088FF84  01800800  00F16968  00F1F8F0  | .h.d.....1...180 |
   +0060  80062024  00F168C0  00010100  F1F8F0D7  | .....1.{....180P |
   +0070  C1D21000  00000000  00000000  | AK.............. |

 Record ID: IOSDMHCB.MHCBTMJB              Length: 0018

   +0000  ACA29DF9  49B6E706  F0F0F0F0  F1F5F0F0  | .s.9..X.00001500 |
   +0010  5CD4C1E2  E3C5D95C                      | *MASTER*         |

 Record ID: IOSDMHCB.MHCBADDL              Length: 0008

   +0000  BCBC2010  01000040                      | .......          |


S530      U/R       000C0001  19:42:52.885939  Unconditional Reserve Sense

 Device: 0180   Channel Path: 38
  U/R Sense issued by: Missing Interrupt Handler
  U/R Sense completion code: 52
```

```
S530      U/R       000C0002  19:43:13.927140  Unconditional Reserve I/O

 Device: 0180
  Recovery I/O issued: Reset Allegiance
  Return code from IOSRRRSV: 04

 Record ID: IOSDRESV.RESS                   Length: 0044

   +0000  D9C5E2E2  05000052  00800000  00000000  | RESS............ |
   +0010  00000000  00000000  | ................ |
   +0020  00000000  00000000  | ................ |
   +0030  00000000  F0000000  80000500  00010760  | ....0..........- |
   +0040  4000C000                                |  .{.             |


S530      U/R       000C0002  17:40:50.336526  Unconditional Reserve I/O

 Device: 0160   Channel Path: 65
  Recovery I/O issued: Sense Path Group ID
  Return code from IOSRRRSV: 00

 Record ID: IOSDICCW.SNID                   Length: 000C

   +0000  C0000111  13214381  AC9EB2D4            | {......a...M     |


S530      U/R       000C0002  22:51:49.169701  Unconditional Reserve I/O

 Device: 0180   Channel Path: 0B
  Recovery I/O issued: Reset Allegiance
  Return code from IOSRRRSV: 00

 Record ID: IOSDICCW.RSTA                   Length: 0020

   +0000  80000000  00000000  00000000  | ................ |
   +0010  00000000  00000000  | ................ |


S530      U/R       000C0002  19:43:39.951626  Unconditional Reserve I/O

 Device: 0180   Channel Path: 38
  Recovery I/O issued: Unconditional Reserve with release
  Return code from IOSRRRSV: 00
```

SYSJES Component Trace

Before Using This Component Trace

This topic assumes you have read:

- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSJES component trace for the JES common coupling services component, also known as JES XCF.

Information	For SYSJES:
Parmlib member	CTnJESxx Default members: CTIJES01, CTIJES02, CTIJES03, CTIJES04
Default tracing	Yes; detailed tracing for sublevel FLOW; minimal tracing of unexpected events for sublevels MSGTRC, USRXIT, XCFEVT
Trace request OPTIONS parameter	None
Buffer	• Default: N/A • Range: N/A • Size set by: MVS system • Change size after IPL: No • Location: In the component address space
Trace records location	Address-space buffer, trace data set
Request of SVC dump	By the component
Trace formatting by IPCS	CTRACE COMP(SYSJES)
Trace format OPTIONS parameter	Yes

Note: To get a complete dump for JES XCF, request also the JES and JES XCF address spaces and data spaces, plus SDATA options RGN, SQA, and CSA.

SYSJES tracing is started during initialization. SYSJES contains 4 sublevel traces, which run concurrently. Each sublevel must be started individually.

The sublevels are:

- **MSGTRC, message tracing**: MSGTRC records message data sent by the IXZXIXSM service. The default tracing for this sublevel is minimal tracing of unexpected events only. You can optionally start and stop detailed MSGTRC tracing. Use the data from this sublevel in conjunction with USRXIT trace data to get information about message data modified by installation exits IXZXIT01 or IXZXIT02.
- **USRXIT, installation exit tracing**: USRXIT records the exit parameter list (SPELL) passed to and returned from installation exits IXZXIT01, IXZXIT02, and IXZXIT03 processing. The default tracing for this sublevel is minimal tracing of unexpected events only. You can optionally start and stop detailed USRXIT tracing. Use the data from this sublevel in conjunction with MSGTRC trace data to get information about message processing through installation exits IXZXIT01, IXZXIT02, and IXZXIT03.
- **FLOW, module footprint tracing**: FLOW records messages and events as they flow through the JES common coupling services component. By default, FLOW is always active and produces detailed tracing.

  Note: IBM recommends that this trace always remain active to record diagnostic data such as errors, system state changes, and processing events.
- **XCFEVT, system event (SYSEVENT) tracing**: XCFEVT records SYSEVENT data processed by the JES common coupling services component. By default, XCFEVT always produces minimal tracing.

Tracing for SYSJES can run all 4 sublevels concurrently. USRXIT and MSGTRC trace only error events by default; you can turn on detailed tracing for these two sublevels.

Requesting a SYSJES Trace

IBM recommends the following when requesting SYSJES TRACING:

- Start and stop the four sublevels for a system all at once in one parmlib member. Request SYSJES component tracing in a CTnJESxx parmlib member which you specify on a TRACE CT command.

  IBM provides two parmlib members, IXZCTION and IXZCTIOF, in SYS1.SAMPLIB as examples of how to start and stop SYSJES sublevels. Copy the members into parmlib, and rename them CTIJESON and CTIJESOF. The CTIJESON parmlib member starts all the sublevels and connects them to the external writer. The CTIJESOF parmlib member stops tracing in all sublevels and disconnects them from the external writer.
- Use the external writer for gathering trace records, because SYSJES tracing produces a large volume of data. Create source JCL for the external writer, using the following guidelines:
  - Code all TRCOUTnn DD statements with a SPACE parameter of at least 10 cylinders to accommodate the volume of SYSJES trace data.
  - For traces larger than 10 cylinders, specify a unique volser for each TRCOUTnn statements if you need to reduce I/O contention on one volume.
  - The data set name defined in the TRCOUT01 DD statement must be unique on each system.
  - Use the IPCS COPYTRC command to merge records from multiple TRCOUTnn DD statements into one data set. See *z/OS MVS IPCS Commands* for information.

Example: Cataloged Procedure for SYSJES

The following example shows an external writer procedure, IXZCTW, that sends SYSJES trace output to trace data sets.

```
//CTWDASD   PROC
//IEFPROC   EXEC  PGM=ITTTRCWR
//SYSPRINT  DD    SYSOUT=A
//TRCOUT01  DD    DSN=SYS1.JESXCF1,VOL=SER=TRACE6,UNIT=DASD,
//          SPACE=(CYL,10),DISP=(NEW,KEEP),DSORG=PS
//TRCOUT02  DD    DSN=SYS1.JESXCF2,VOL=SER=TRACE7,UNIT=DASD,
//          SPACE=(CYL,10),DISP=(NEW,KEEP),DSORG=PS
```

- If you are tracing in a sysplex environment, the data set names on TRCOUTnn DD statements must be unique throughout the sysplex. An ENQUEUE error results if the data set names are not unique.

CTnJESxx Parmlib Member

The following table indicates the parameters you can specify on a CTnJESxx parmlib member.

Parameters	Allowed on CTnJESxx?
ON or OFF	Yes
ASID	No
JOBNAME	No
BUFSIZE	No
OPTIONS	No
SUB	Yes

Parameters	Allowed on CTnJESxx?
PRESET	No
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON or OFF	One is required
nnnnK or nnnnM	No
COMP	Required
SUB	Yes
PARM	Yes

Parameters	Allowed on TRACE CT for Writer?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	No
JOBNAME	No
OPTIONS	No
WTR	Yes

Examples of Requesting SYSJES Traces

Example 1: Start SYSJES Tracing with the CTIJESON Member

The following example shows the CTIJESON parmlib member supplied in SYS1.SAMPLIB to start tracing for one system:

```
TRACEOPTS
WTRSTART(IXZCTW) WRAP
SUB(MSGTRC)
ON
WTR(IXZCTW)
SUB(USRXIT)
ON
WTR(IXZCTW)
SUB(FLOW)
ON
WTR(IXZCTW)
SUB(XCFEVT)
ON
WTR(IXZCTW)
```

Example 2: Stop SYSJES Tracing with the CTIJESOF Member

The following example shows the CTIJESOF parmlib member supplied in SYS1.SAMPLIB to stop tracing for one system:

```
TRACEOPTS
SUB(MSGTRC)
OFF
SUB(USRXIT)
OFF
SUB(FLOW)
OFF
SUB(XCFEVT)
OFF
```

Stop tracing with the following command:

```
TRACE CT,OFF,COMP=SYSJES,PARM=CTIJESOF
```

Then specify the following command to stop the external writer (assuming IXZCTW is the membername of the source JCL for the external writer):

```
TRACE CT,WTRSTOP=IXZCTW
```

Requesting a SYSJES Trace for Problems During Initialization

Use this procedure only when requested to by the IBM Support Center. The procedure requests SYSJES tracing for JES XCF problems occurring during JES subsystem initialization. The procedure consists of using the default parmlib members CTIJES01, CTIJES02, and CTIJES04 to request tracing of these SYSJES sublevels. Note that parmlib member CTIJES03 contains module footprint tracing that is active, by default, on your system. Therefore, you do not need to take any action to modify this trace.

Activating all four traces can negatively impact system performance because of the heavy volume of trace data produced. For that reason, you should only use this procedure when requested by IBM, and you should not leave this full tracing on. The identical parmlib members are supplied with tracing set off, since you should only run with full tracing at IBM's request. The default contents of parmlib members CTIJES01, CTIJES02, and CTIJES04 is:

```
TRACEOPTS
OFF
```

The default contents of parmlib members CTIJES03 is:

```
TRACEOPTS
ON
```

IBM recommends that you keep this tracing sublevel on at all times.

1. Modify the CTIJES01, CTIJES02, and CTIJES04 Parmlib Members to Turn Tracing On: In parmlib members CTIJES01, CTIJES02, and CTIJES04, alter the parmlib members to return sublevel tracing on and connect the sublevel to the external writer. The parmlib members are supplied with tracing off and no connection to the writer.

When you initialize the JES subsystem with the modified parmlib members, full tracing for JES XCF starts automatically.

Example: Turning on Tracing in a CTIJESxx Member

The following is an example of the CTIJESxx parmlib member after having been modified for gathering trace data during JES subsystem initialization at the direction of the IBM Support Center. The member turns tracing on for the sublevel and connects the sublevel to the external writer.

```
TRACEOPTS
WTRSTART(IXZCTW)
ON
WTR(IXZCTW)
```

2. Create a CTIJESOF Parmlib Member to Stop SYSJES Tracing: Use the CTIJESOF parmlib member to stop the full SYSJES tracing turned on during initialization and to disconnect them from the external writer.

3. Stop SYSJES Tracing after Initialization Tracing is Complete: Enter a TRACE CT operator command referencing the CTIJESOF parmlib member on the console with master authority as follows:

```
TRACE CT,OFF,COMP=SYSJES,PARM=CTIJESOF
```

4. Remodify the CTIJES01, CTIJES02, and CTIJES04 Parmlib Members to Return to Default Tracing: In parmlib members CTIJES01, CTIJES02, and CTIJES04, alter the parmlib member to return sublevel tracing to off.

Example: Return to Default in a CTIJESxx Member

The following example shows the CTIJES01, CTIJES02,and CTIJES04 parmlib members after having been returned to their original contents, with tracing set off.

```
TRACEOPTS
OFF
```

Formatting a SYSJES Trace

Format the trace with an IPCS CTRACE COMP(SYSJES) subcommand. To format SYSJES tracing, you must:

- Enter the CTRACE command for SYSJES once for each of the four sublevels you wish to format. See “Format SYSJES Sublevel Information”.
- Specify SYSJES options on the OPTIONS parameter. See “OPTIONS Parameter for Formatting a SYSJES Trace” on page 11-66.
- Merge the output from the different sublevels requested. See “Merging SYSJES Information From Sublevels” on page 11-66.

For SYSJES traces, use the IPCS MERGE subcommand to display traces that are not likehead in timestamp order.

Format SYSJES Sublevel Information

You must enter the CTRACE command separately for each SYSJES sublevel you wish to format. For example, to request formatting of SYSJES trace data for sublevels MSGTRC and USRXIT, you would enter the following two commands:

```
CTRACE COMP(SYSJES) SUB((USRXIT)) FULL
CTRACE COMP(SYSJES) SUB((MSGTRC)) FULL
```

These examples would yield tracing without any options requested.

OPTIONS Parameter for Formatting a SYSJES Trace

IBM might request that you enter options for SYSJES tracing. You can specify options for SYSJES tracing on the OPTIONS parameter of the CTRACE command. The options include:

- Options valid for all sublevels:
  - MSGTOKEN=*msgtoken*
  - REQTOKEN=*reqtoken*
  - MSGBUF=*msgbuf*
  - CTCR=*ctcr*
- Options valid for the FLOW sublevel only:
  - MODID=*id*
  - MODFLOW
  - MSGFLOW
- Options valid for the MSGTRC sublevel only:
  - SEND
  - RECEIVE
- Options valid for the USRXIT sublevel only:
  - EXIT1
  - EXIT3
  - EXIT2

Merging SYSJES Information From Sublevels

Because SYSJES can run four sublevel traces simultaneously, you will need to merge the data for a complete chronological picture of SYSJES trace data. For example, to merge JESXCF trace data, from all four sublevels, enter the following command:

```
MERGE
CTRACE COMP(SYSJES) SUB((USRXIT)) FULL
CTRACE COMP(SYSJES) SUB((MSGTRC)) FULL
CTRACE COMP(SYSJES) SUB((XCFEVT)) FULL
CTRACE COMP(SYSJES) SUB((FLOW)) FULL
MERGEEND
```

You can write an IPCS CLIST to issue the CTRACE command for the four sublevels and merge the output automatically. See *z/OS MVS IPCS Customization* for information on writing a CLIST.

Output from a SYSJES Trace

The following is an example of merged output from the four SYSJES sublevel traces with the FULL parameter specified:

```
********** MERGED TRACES ***********
01. CTRACE comp(sysjes) sub((flow)) full
02. CTRACE comp(sysjes) sub((usrxit)) full
03. CTRACE comp(sysjes) sub((msgtrc)) full
04. CTRACE comp(sysjes) sub((xcfevt)) full
```

```
 COMPONENT TRACE FULL FORMAT
 SYSNAME(SY1)
 COMP(SYSJES)  SUBNAME((FLOW))
```

```
 COMPONENT TRACE FULL FORMAT
 SYSNAME(SY1)
 COMP(SYSJES)  SUBNAME((USRXIT))
```

```
 COMPONENT TRACE FULL FORMAT
 SYSNAME(SY1)
 COMP(SYSJES)  SUBNAME((MSGTRC))
```

```
 COMPONENT TRACE FULL FORMAT
 SYSNAME(SY1)
 COMP(SYSJES)  SUBNAME((XCFEVT))

 **** 10/25/93
      MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
      --------  --------  ---------------  -----------
 04.  XCFEVT    00000030  16:53:54.876999  XCF event message buffer
    HOMEASID 0013     JOBNAME. JESXCF   HOMETCB@ 007EA6F8
    CPUID... FF170945 30900000          CTCR@... 03062460 MEMBER.. SY1
                      SYSNAME. SY1
    MSGBUF@. 20000014 ALET.... 0101001B REQTOKEN 00000001 20000014
    MSGTOKEN 00000001 20000014
    +0000  E8C9E7C5 D5400100 00000001 00000001 | YIXEN .......... |
    +0010  E2E8E2E9 D1C5E2F3 E2E8E2C5 E5C5D5E3 | SYSZJES3SYSEVENT |
    +0020  40404040 40404040 E2E8E2C5 E5C5D5E3 |         SYSEVENT |
    +0030  40404040 40404040 E2E8E2E9 D1C5E2F3 |         SYSZJES3 |
    +0040  E2E8E2C5 E5C5D5E3 40404040 40404040 | SYSEVENT         |
    +0050  E2E8E2C5 E5C5D5E3 40404040 40404040 | SYSEVENT         |
    +0060  10001000 80000108 01140000 00000000 | ................ |
    +0070  00000000 A849C5A7 AE08EC01 00000000 | ....y.Ex........ |
    +0080  00000000 00000000 00000000 00000000 | ................ |
                              .
                              .
                              .
```

```
**** 10/25/93
     MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
     --------  --------  ---------------  -----------
02.  PREXIT3   00000007  16:53:55.303610  SPELL prior to Exit 3 processing
   HOMEASID 0017     JOBNAME. JES3     HOMETCB@ 007FD2B8
   CPUID... FF170945 30900000          CTCR@... 03062460 MEMBER.. SY1
                     SYSNAME. SY1
   XPLBUF@. 03368000
   +0000  E95BE7D7 D3400101 C9E7E9E7 C9E3F0F3 | Z$XPL ..IXZXIT03 |
   +0010  40404040 40404040 40000000 80000000 |          ....... |
   +0020  00000000 00000000 00000000 00000000 | ................ |
   +0030  00000054 00000038 E2E8E2E9 D1C5E2F3 | ........SYSZJES3 |
   +0040  E2E8F140 40404040 40404040 40404040 | SY1              |
   +0050  00000000                            | ....             |
02.  POSTXIT3  00000008  16:53:55.306462  SPELL after Exit 3 processing
   HOMEASID 0017     JOBNAME. JES3     HOMETCB@ 007FD2B8
   CPUID... FF170945 30900000          CTCR@... 03062460 MEMBER.. SY1
                     SYSNAME. SY1
   XPLBUF@. 03368000
   +0000  E95BE7D7 D3400101 C9E7E9E7 C9E3F0F3 | Z$XPL ..IXZXIT03 |
   +0010  40404040 40404040 40000000 80000000 |          ....... |
   +0020  00000000 00000000 00000000 00000000 | ................ |
   +0030  00000054 00000038 E2E8E2E9 D1C5E2F3 | ........SYSZJES3 |
   +0040  E2E8F140 40404040 40404040 40404040 | SY1              |
   +0050  00000000                            | ....             |
**** 10/25/93
     MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
     --------  --------  ---------------  -----------
03.  INDATA    00000011  16:55:17.318441  input message data
   HOMEASID 0017     JOBNAME. JES3     HOMETCB@ 007FD2B8
   CPUID... FF170945 30900000          CTCR@... 03062460 MEMBER.. SY1
                     SYSNAME. SY1
   MSGBUF@. 1FFFF814 ALET.... 0102001B REQTOKEN 00000001 1FFFF814
   MSGTOKEN 00000001 1FFFF814
   +0000  E8C9E7C5 D5400100 00000001 00000001 | YIXEN .......... |
   +0010  E2E8E2E9 D1C5E2F3 E2E8F240 40404040 | SYSZJES3SY2      |
   +0020  40404040 40404040 E2E8E2D1 C5E24040 |         SYSJES   |
   +0030  C3D6D5E2 C5D9E540 E2E8E2E9 D1C5E2F3 | CONSERV SYSZJES3 |
   +0040  E2E8F140 40404040 40404040 40404040 | SY1              |
   +0050  E2E8E2D1 C5E24040 C3D6D5E2 C5D9E540 | SYSJES  CONSERV  |
   +0060  20001000 20000100 01148000 00000000 | ................ |
   +0070  00000000 A849C5F6 19827F03 00000000 | ....y.E6.b".....  |
   +0080  00000000 007FD2B8 00FC1D00 00000000 | ....."K......... |
   +0090  00000214 00000000 00000000 00000001 | ................ |
   +00A0  1FFFF814 00000000 00000000 00000000 | ..8............. |
                            .
                            .
                            .
```

```
02.  EXIT1ERR  00000003  16:55:17.474658  SPELL error Exit 1 processing
   HOMEASID 0017     JOBNAME. JES3     HOMETCB@ 007FD2B8
   CPUID... FF170945 30900000          CTCR@... 03062460 MEMBER.. SY1
                     SYSNAME. SY1
   XPLBUF@. 1FFFF414
   MSGBUF@. 1FFFF814 ALET.... 0102001B REQTOKEN 00000001 1FFFF814
   MSGTOKEN 00000001 1FFFF814
   +0000  E95BE7D7 D3400100 C9E7E9E7 C9E3F0F1 | Z$XPL ..IXZXIT01 |
   +0010  40404040 40404040 40000000 20000000 |          ....... |
   +0020  00000000 00000000 00000000 00000000 | ................ |
   +0030  000001A0 00000038 E2E8E2E9 D1C5E2F3 | ........SYSZJES3 |
   +0040  E2E8F240 40404040 40404040 40404040 | SY2              |
   +0050  E2E8E2D1 C5E24040 C3D6D5E2 C5D9E540 | SYSJES  CONSERV  |
   +0060  E2E8E2E9 D1C5E2F3 E2E8F140 40404040 | SYSZJES3SY1      |
   +0070  40404040 40404040 E2E8E2D1 C5E24040 |         SYSJES   |
   +0080  C3D6D5E2 C5D9E540 00000100 1FFFF4B4 | CONSERV ......4. |
   +0090  00000000 00000000 0000E300 00000000 | ..........T..... |
   +00A0  E3C8C9E2 40C9E240 C1D540C1 E2E8D5C3 | THIS IS AN ASYNC |
   +00B0  C8C1C3D2 40D4C5E2 E2C1C7C5 40404040 | HACK MESSAGE     |
   +00C0  40404040 40404040 40404040 40404040 |                  |
   +00D0  40404040 40404040 40404040 40404040 |                  |
   +00E0  40404040 40404040 40404040 40404040 |                  |
   +00F0  40404040 40404040 40404040 40404040 |                  |
   +0100  40404040 40404040 40404040 40404040 |                  |
   +0110  40404040 40404040 40404040 40404040 |                  |
   +0120  40404040 40404040 40404040 40404040 |                  |
   +0130  40404040 40404040 40404040 40404040 |                  |
   +0140  40404040 40404040 40404040 40404040 |                  |
   +0150  40404040 40404040 40404040 40404040 |                  |
   +0160  40404040 40404040 40404040 40404040 |                  |
   +0170  40404040 40404040 40404040 40404040 |                  |
   +0180  40404040 40404040 40404040 40404040 |                  |
   +0190  40404040 40404040 40404040 40404040 |                  |
03.  SEND      00000010  16:55:17.568558  message queued to be sent
   HOMEASID 0013     JOBNAME. JESXCF   HOMETCB@ 007EA6F8
   CPUID... FF170945 30900000          CTCR@... 03062460 MEMBER.. SY1
                     SYSNAME. SY1
   MSGBUF@. 1FFFF814 ALET.... 0102001B REQTOKEN 00000001 1FFFF814
   MSGTOKEN 00000001 1FFFF814
   +0000  E8C9E7C5 D5400100 00000001 00000001 | YIXEN .......... |
   +0010  E2E8E2E9 D1C5E2F3 E2E8F240 40404040 | SYSZJES3SY2      |
   +0020  40404040 40404040 E2E8E2D1 C5E24040 |         SYSJES   |
   +0030  C3D6D5E2 C5D9E540 E2E8E2E9 D1C5E2F3 | CONSERV SYSZJES3 |
   +0040  E2E8F140 40404040 40404040 40404040 | SY1              |
   +0050  E2E8E2D1 C5E24040 C3D6D5E2 C5D9E540 | SYSJES  CONSERV  |
                           .
                           .
                           .
```

The following is an example of merged output from the four SYSJES sublevel traces with the TALLY parameter specified. Use the TALLY report to look at event identifiers in the trace output.

```
 COMPONENT TRACE TALLY REPORT
 SYSNAME(SY1)
 COMP(SYSJES)  SUBNAME((FLOW))
     TRACE ENTRY COUNTS AND AVERAGE INTERVALS (IN MICROSECONDS)
 FMTID     COUNT     INTERVAL      MNEMONIC  DESCRIBE
 --------  --------  ------------  --------  --------------------------------
 00000020        29     6,902,004  MSGFLOW   module flow for this message
 00000021       114     1,755,686  MODFLOW   module flow
 0000002E         3     2,631,973  MSGERR    module error flow
 0000002F         7    27,207,632  MODERR    module error flow
Total trace entries:      153
```

```
 COMPONENT TRACE TALLY REPORT
 SYSNAME(SY1)
 COMP(SYSJES)  SUBNAME((MSGTRC))
     TRACE ENTRY COUNTS AND AVERAGE INTERVALS (IN MICROSECONDS)
 FMTID     COUNT     INTERVAL      MNEMONIC  DESCRIBE
 --------  --------  ------------  --------  --------------------------------
 00000010        13    17,076,718  SEND      message queued to be sent
 00000011        17    12,821,783  INDATA    input message data
 00000012         0                SENDBUF   buffer sent by XCF
 00000013         0                SENDQERR  Sendq queueing error
 00000014         0                SBUFERR   Send error from XCF
 00000018        13    17,061,037  RECEIVE   msg dequeued from receive
 00000019        10    30,944,882  OUTDATA   output message data
 0000001A        26     8,192,076  RECVBUF   buffer received from XCF
 0000001B         0                RBUFERR   Receive error from XCF
 0000001C         0                RECVQERR  Receiveq queueing error
Total trace entries:       79
```

```
 COMPONENT TRACE TALLY REPORT
 SYSNAME(SY1)
 COMP(SYSJES)  SUBNAME((XCFEVT))
     TRACE ENTRY COUNTS AND AVERAGE INTERVALS (IN MICROSECONDS)
 FMTID     COUNT     INTERVAL      MNEMONIC  DESCRIBE
 --------  --------  ------------  --------  --------------------------------
 00000030        27    35,794,857  XCFEVT    XCF event message buffer
 00000031         0                XCFERR    XCF event message error
Total trace entries:       27
```

```
 COMPONENT TRACE TALLY REPORT
 SYSNAME(SY1)
 COMP(SYSJES)  SUBNAME((USRXIT))
     TRACE ENTRY COUNTS AND AVERAGE INTERVALS (IN MICROSECONDS)
 FMTID     COUNT     INTERVAL      MNEMONIC  DESCRIBE
 --------  --------  ------------  --------  --------------------------------
 00000001         1                PREXIT1   SPELL prior to Exit 1 processing
 00000002         0                POSTXIT1  SPELL after Exit 1 processing
 00000003         1                EXIT1ERR  SPELL error Exit 1 processing
 00000004         1                PREXIT2   SPELL prior to Exit 2 processing
 00000005         0                POSTXIT2  SPELL after Exit 2 processing
 00000006         1                EXIT2ERR  SPELL error Exit 2 processing
 00000007        14    60,329,405  PREXIT3   SPELL prior to Exit 3 processing
 00000008        14    60,374,425  POSTXIT3  SPELL after Exit 3 processing
 00000009         0                EXIT3ERR  SPELL error Exit 3 processing
Total trace entries:       32
```

SYSjes2 Component Trace

Before Using This Component Trace

This topic assumes you have read:
- "Planning for Component Tracing" on page 11-3
- "Obtaining a Component Trace" on page 11-10
- "Viewing the Component Trace Data" on page 11-23

The following summarizes information for requesting a SYSjes2 component trace for the JES2 subsystem. For ease of explanation here, this component trace is referred to as **SYS*jes2*** although you might need to replace ***jes2*** with the name you assigned to your JES2 subsystem (primary or secondary). For example, to obtain trace information for JESA, a name you might have used to name your secondary JES2 in a poly-JES environment, use *SYSJESA* as the component name.

Information	For SYSjes2:
Parmlib member	n/a
Default tracing	Yes; full tracing for sublevels JQE and JOE
Trace request OPTIONS parameter	None
Buffer	• Default: N/A • Range: N/A • Size set by: JES2 • Change size after IPL: No • Location: In the component address space
Trace records location	Address-space buffer
Request of SVC dump	By the component, or DUMP or SLIP
Trace formatting by IPCS	CTRACE COMP(SYSjes2)
Trace format OPTIONS parameter	Yes

SYSjes2 tracing is started automatically during initialization. SYSjes2 contains 2 sublevel traces, which run continuously and concurrently.

The sublevels are:
- **JQE service tracing**: JQE records all job queue service calls (to include: $QADD, $QPUT, $QREM, $QMOD, $QJIX, $GETJLOK, $QRBDCHK, $QBUSY, $GETLOKW, $FREJLOK, and $FRELOKW).
- **JOE service tracing**: JOE records all job output element service calls (to include: $#ADD, $#PUT, $#REM, $#MOD, $#RBDCHK, $#BUSY, $#GET, and $#CAN).

Requesting a SYSjes2 Trace

You need not take any action to request a SYSjes2 trace. The trace is active whenever your JES2 subsystem is in control.

Formatting SYSjes2 Sublevel Trace Information

You must enter the CTRACE command separately for each SYSJES sublevel you wish to format. For example, to request formatting of SYSJES trace data for sublevels JQE and JOE, you would enter the following two commands:

```
CTRACE COMP(SYSjes2) SUB((JQE)) FULL
CTRACE COMP(SYSjes2) SUB((JOE)) FULL
```

Merging SYSjes2 Information From Sublevels

Because SYSjes2 runs two sublevel traces simultaneously, you will need to merge the data for a complete chronological picture of SYSjes2 trace data. To merge SYSjes2 trace data, enter the following command string:

```
MERGE
CTRACE COMP(SYSjes2) SUB((JQE)) FULL
CTRACE COMP(SYSjes2) SUB((JOE)) FULL
MERGEEND
```

You can write an IPCS CLIST to issue the CTRACE command for both sublevels and merge the output automatically. See *z/OS MVS IPCS Customization* for information on writing a CLIST.

Output from a SYSjes2 Trace

The output from the SYSjes2 trace contains a maximum of 500 trace entries presented in a wrapping (or rolling) trace format. That is, once the trace table is filled with 500 entries, the next entries (501, 502, 503,...) overwrite entries 1, 2, 3... in a continuous wrapping manner.

The following is an example of merged output from both SYSjes2 sublevel traces with the **FULL** parameter specified:

```
********** MERGED TRACES ***********
01. CTRACE comp(sysjes2) sub((jqe)) full
02. CTRACE comp(sysjes2) sub((joe)) full
```

The following is an example of merged output from both SYSjes2 sublevel traces with the **SHORT** parameter specified:

```
**********************short format screen ********************

 COMPONENT TRACE SHORT FORMAT
 COMP(SYSjes2) SUBNAME((JQE))
 **** 07/11/94

 SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
 -------   --------  --------  ---------------  -------------

 MVS1      JQE       00000200  14:33:35.162400  $QADD

 MVS1      JQE       00000204  14:33:35.162404  $QJIX (ALLOC new number)

 MVS1      JQE       00000200  14:33:36.174242  $QADD

------------------------------------------------------------------------
```

The following is an example of merged output from both SYSjes2 sublevel traces with the **FULL** parameter specified. Use mapping macro, $ROTT (rolling trace table), to map the fields presented in this trace.

```
********************full format screen ******************************

IPCS OUTPUT STREAM ------------------------------------------ Line 0 Co
 ****************************** TOP OF DATA ***************************


  COMPONENT TRACE FULL FORMAT
  COMP(SYSjes2) SUBNAME((JQE))
  **** 07/11/94

  SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
  -------   --------  --------  ---------------  -------------

  MVS1      JQE       00000200  14:33:35.162400  $QADD

     03B06C20  00000000  00000064  FF200100  | ..%............. |
     00000000  0000                          | ......            |
  MVS1      JQE       00000204  14:33:35.162404  $QJIX (ALLOC new number

     03B06C20  00000001  00000064  20200100  | ..%............. |
     04000000  0000                          | ......            |
  MVS1      JQE       00000200  14:33:36.174242  $QADD

------------------------------------------------------------------------
```

The following is an example of merged output from both SYSjes2 sublevel traces with the **TALLY** parameter specified:

```
**************************tally format screen *************************
 COMPONENT TRACE TALLY REPORT
 COMP(SYSjes2) SUBNAME((JQE))
     TRACE ENTRY COUNTS AND AVERAGE INTERVALS (IN MICROSECONDS)

 FMTID     COUNT     INTERVAL      MNEMONIC  DESCRIBE
 --------  --------  ------------  --------  ---------------------------
 00000200        12    96,278,898  JQE       $QADD
 00000201         0                JQE       $QPUT
 00000202         0                JQE       $QREM
 00000203        30    36,473,953  JQE       $QMOD
 00000204        12    96,278,898  JQE       $QJIX (ALLOC new number)
 00000205         0                JQE       $QJIX (SWAP job numbers)
 00000206         5   273,117,249  JQE       $GETJLOK
 00000207         5   274,068,170  JQE       $FREJLOK
   .
   .
   .
  Total trace entries:       89
  ****************************** END OF DATA ***************************
```

SYSLLA Component Trace

Before Using This Component Trace

This topic assumes you have read:

- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSLLA component trace for library lookaside (LLA) of contents supervision.

Information	For SYSLLA:
Parmlib member	None
Default tracing	Yes; always on when LLA is running
Trace request OPTIONS parameter	None
Buffer	• Default: N/A • Range: N/A • Size set by: MVS system • Change size after IPL: No • Location: In the component address space
Trace records location	Address-space buffer
Request of SVC dump	By the component
Trace formatting by IPCS	CTRACE COMP(SYSLLA)
Trace format OPTIONS parameter	None

Requesting a SYSLLA Trace

The trace runs whenever LLA is in control. No actions are needed to request it.

Formatting a SYSLLA Trace

Format the trace with an IPCS CTRACE COMP(SYSLLA) subcommand. The subcommand has no OPTIONS values.

SYSLOGR Component Trace

Before Using This Component Trace

This topic assumes you have read:

- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSLOGR component trace for the system logger component.

Information	For SYSLOGR:
Parmlib member	CTnLOGXX No default member
Default tracing	Yes; minimal; unexpected events
Trace request OPTIONS parameter	In CTnLOGxx and REPLY for TRACE command
Buffer	• Default: 2MB • Range: 2MB - 2047MB • Size set by: CTnLOGxx parmlib member or REPLY for TRACE CT command. • Change size after IPL: Yes • Location: system logger trace data space
Trace records location	Address-space buffer; system logger trace data space, trace data set
Request of SVC dump	By DUMP or SLIP command

Information	For SYSLOGR:
Trace formatting by IPCS	CTRACE COMP(SYSLOGR)
Trace format OPTIONS parameter	Yes

SYSLOGR tracing is started during initialization.

Getting a Dump of System Logger Information

Use the following examples to obtain the appropriate diagnostic information for system logger. The amount of information requested in the dumps may be very large. You may need to set your MAXSPACE on the CHNGDUMP setting to 999meg before obtaining the logger dumps

```
CD SET,SDUMP,MAXSPACE=999M
```

1. For all types of logstreams, always include the following:
   a. The IXGLOGR (Logger) asid and the data spaces associated with the IXGLOGR asid through the DSPNAME parm. These will be dumped using the JOBNAME= parm.
   b. The trace data space SYSLOGR0, will be included in the dump if DSNAME=('IXGLOGR'.*) is specified in the rpelyfor the dump command.

      Note: If you are running OS/390 Release 2 or lower, Logger will not have a SYSLOGR* data space for tracing, in which case the DSPNAME=('IXGLOGR'.*) option can be omitted.

```
DUMP  COMM=(dump system logger with component trace)
* rr IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
rr,JOBNAME=(IXGLOGR),CONT
ss,DSPNAME=('IXGLOGR'.SYSLOGR*),CONT
tt,SDATA=(COUPLE,ALLNUC,LPA,LSQA,PSA,RGN,SQA,SUM,
          TRT,CSA,GRSQ,XESDATA),END
```

2. When using CF list structure based logstreams, include the following
   a. The XCF asid and trace data spaces. These will be dumped using the JOBNAME= parm and DSPNAME parm.
   b. The XES STRUCTURE data. This is dumped using the STRLIST= parameter and by specifying the structure name. structure_name is the affected STRUCTURE name.

      Note: Be sure to allocate adequate DUMPSPACE() as defined in your CF definition in the CFRM policy. If you do not allocate adequate space, all or part of the STRUCTURE will NOT be dumped.
   c. In the case of “loss of data” or “block not found”, dumping the OFFLOAD data sets using IDCAMS is a good idea.

```
DUMP COMM=(dump system logger with xes and structure data)
* rr IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
rr,JOBNAME=(IXGLOGR,XCFAS),CONT
ss,DSPNAME=('XCFAS'.*,'IXGLOGR'.SYSLOGR*),CONT
tt,SDATA=(COUPLE,ALLNUC,LPA,LSQA,PSA,RGN,SQA,SUM,
          TRT,CSA,GRSQ,XESDATA),CONT
uu,STRLIST=(STRNAME=structure_name,LOCKENTRIES,ACC=NOLIM,
    (LISTNUM=ALL,ADJUNCT=DIRECTIO,ENTRYDATA=UNSERIALIZE)),END
```

3. When system logger dumps are needed on multiple systems in the sysplex, include the REMOTE parameter.

```
DUMP COMM=(dumps for system logger around the sysplex)
* rr IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
rr,JOBNAME=(IXGLOGR,XCFAS),CONT
ss,DSPNAME=('XCFAS'.*,'IXGLOGR'.SYSLOGR*),CONT
```

```
tt,SDATA=(COUPLE,ALLNUC,LPA,LSQA,PSA,RGN,SQA,SUM,
           TRT,CSA,GRSQ,XESDATA),CONT
uu,STRLIST=(STRNAME=structure_name,LOCKENTRIES,ACC=NOLIM,
     (LISTNUM=ALL,ADJUNCT=DIRECTIO,ENTRYDATA=UNSERIALIZE)),CONT
vv,REMOTE=(SYSLIST=*)'XCFAS','IXGLOGR'),DSPNAME,SDATA),END
```

4. Other diagnostic considerations

   For CTRACE, we recommend a 10meg buffer size. The default is 2meg. See the CTnLOGxx member, described in this document.

   Remember that much of the data that system logger uses is persistent across an IPL. That means that if this data is corrupted and adversely affects system logger, an IPL will not correct the problem. In the case of a persistent system logger failure, you can FORCE the IXGLOGR address space. Prior to doing this you should bring down all of the applications connected. Then issue the FORCE command (

   ```
   FORCE IXGLOGR,ARM
   ```

) and restart system logger using the supplied procedure in SYS1.PROCLIB (IXGLOGRS). (

   ```
   S IXGLOGRS
   ```

)

   If

   ```
   FORCE IXGLOGR,ARM
   ```

   does NOT resolve the situation, an IPL is not likely to either. This is the time to take a dump if one was not already taken by system logger.

   Notes:

   a. A CICS dump will not dump the IXGLOGR address space. Connect to a new (non-corrupted) LOGSTREAM. This will result in a LOSS OF DATA for some applications such as CICS, forcing them to INITIAL START. However, this may be the only way to get the application restarted. Connecting to a new logstream (of a different name) will allow the corrupted data to remain in the structure for diagnostic review later.
   b. In preparation for connecting to this new LOGSTREAM, you may want to define an unused LOGSTREAM to each STRUCTURE during setup.

   If running CICS, always run with the following SLIP:

   ```
   SLIP SET,COMP=1C5,REASON=804,
     STRLIST=(STRNAME=strname1,LOCKE,ACC=NOLIM,
     (LNUM=ALL,EDATA=SER,ADJ=CAP)),
     JL=(XCFAS,IXGLOGR),DN=("XCFAS".*,"IXGLOGR".*),
     SD=(COUPLE,ALLNUC,LPA,LSQA,PSA,RGN,SQA,TRT,CSA,GRSQ,
       XESDATA,SUM),
     SUMLIST=(13R?-7FFF,13R?+7FFF),END
   ```

   Note: You might add a JOBNAME=DFH* to limit SLIP to CICS. A RSN804 is a “block not found”, which is always bad for CICS but not necessarily so for other applications. Setting this SLIP will cause system logger to dump on all RSN804s in CICS.

5. Frequent stumbling blocks

   a. OFFLOAD data sets must be VSAM SHAREOPTIONS(3,3) unless you are in a MONOPLEX.

b. After OW33261, system logger recommends for performance reasons using 24K CI size for OFFLOAD data sets. Staging data sets must remain at 4K CI size. Code your ACS routines appropriately.

c. Size of XES structures. “Bigger is not always better.” Follow exploiting application recommendations.

d. Allow for OFFLOAD directory extents. Reference “Format Utility for Couple Data Sets” in *z/OS MVS Setting Up a Sysplex*.

e. System logger uses HSM services to recall (ARCHRCAL) and to delete (ARCHDEL) offload data sets. HSM contention or a wait for a WTOR such as ARC0055A can hang all of the log streams that require the system logger allocation task.

Requesting a SYSLOGR Trace

Specify options for requesting a SYSLOGR component trace in a CTnLOGxx parmlib member or on the reply for a TRACE CT command.

You can change options for SYSLOGR tracing while the trace is running. You can change both the options and the trace data space buffer size for SYSLOG trace while the trace is running. However, if the SYSLOGR trace has not been connected to an external writer and you are reducing the size of the trace data space buffer, you **must** dump the contents of the buffer (see “Getting a Dump of System Logger Information” on page 11-75) **before** reducing the buffer size if this data is important for debugging. Trace data in the trace data space is discarded when the buffer size is reduced.

Note that if the trace is being turned off (either via a TRACE CT,OFF command or a CTnLOGxx parmlib member) and the current trace data space buffer size is greater than the 2MB default, the buffer size automatically reverts back to 2MB. If the SYSLOGR trace is not connected to an external writer, the trace data **must** be dumped **before** turning the trace off to avoid loss of data.

CTnLOGxx Parmlib Member

The following table indicates the parameters you can specify in a CTnLOGxx parmlib member.

Parameters	Allowed on CTnLOGxx?
ON or OFF	Yes
ASID	Yes
JOBNAME	No
BUFSIZE	Yes
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

Example of CTnLOGxx Parmlib Member: The statements in the following CTnLOGxx parmlib member example specify a 4MB trace buffer. All system logger trace records will be included in the trace output:

```
TRACEOPTS
  ON
  BUFSIZE(4M)
  OPTIONS('ALL')
```

The statements in the following CTxLOGxx example specify a 2MB trace buffer, with tracing of logstream functional request processing for logstreams SAMPLE1 and SAMPLE2 in ASID 04. In addition, an external writer, EXTWTR, will be started, and SYSLOGR will be connected to the external writer:

```
TRACEOPTS
  WTRSTART(EXTWTR)
  ON
  BUFSIZE(2M)
  ASID(04)
  WTR(EXTWTR)
  OPTIONS('LOGSTRM','STRMNAME=(SAMPLE1,SAMPLE2)')
```

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON, nnnnM, or OFF	Yes
COMP	Required
SUB	No
PARM	Yes

If you reduce the size of the trace data space buffer and the SYSLOG trace has not been connected to an external writer, you **must** dump the contents of the buffer (see “Getting a Dump of System Logger Information” on page 11-75) **before** reducing the buffer size if this data is important for debugging. Trace data in the trace data space buffer is discarded when the buffer size is reduced.

If the trace is being turned off (either via a TRACE CT,OFF command or a CTnLOGxx parmlib member) and the current trace data space buffer size is greater than the 2MB default, the buffer size automatically reverts back to 2MB. If the SYSLOGR trace is not connected to an external writer, the trace data **must** be dumped **before** turning the trace off to avoid loss of data.

Parameters	Allowed on REPLY for Trace?
ASID	Yes
JOBNAME	No
OPTIONS	Yes
WTR	Yes

OPTIONS Parameter

The values for the OPTIONS parameter for the CTnLOGxx parmlib member and reply for a TRACE command, in alphabetical order, are:

Option	DeFault	Subparameters
ALL	No	No
CONNECT	Yes	No
COMPERR	Min	No

Option	DeFault	Subparameters
DATASET	Yes	No
INVENTRY	No	No
LOCBUFF	Yes	No
LOGSTRM	Yes	No
MISC	Yes	No
RECOVERY	Yes	No
SERIAL	Yes	No
STRMNAME	No	Logstream
STORAGE	No	No

ALL
Traces all system logger events.

ASID
Traces events for only the specified address space identifiers (ASID).

COMPERR
Traces internal system logger errors.

CONNECT
Traces list structure connections, disconnections, rebuild and event exit processing.

DATASET
Traces log stream data set allocation and management.

INVENTRY
Traces log stream and structure definition and deletion processing as well as LOGR policy processing.

LOCBUFF
Traces system logger local buffer management.

LOGSTRM
Traces log stream functional request processing.

MISC
Traces system logger internal miscellaneous services.

RECOVERY
Traces system logger component recovery, detecting abnormal conditions during processing.

SERIAL
Traces system logger serialization services.

STORAGE
Traces system logger storage management.

STRMNAME
Traces events for only the specified log streams. If you specify STRMNAME, the specified log streams filter the CONNECT, LOGSTRM, INVENTRY, and DATASET options. The STRMNAME parameter must be specified STRMNAME=(*strmname1*). If you specify more than one log stream, STRMNAME must be specified STRMNAME=(*strmname1*,*strmname2*). A maximum of eight log stream names can be specified.

Note that the system does not verify the log stream names specified.

Example of Requesting SYSLOGR Trace

Example: CTnLOGxx Member

The member requests CONNECT and DATASET processing.

```
TRACEOPTS
  ON
  OPTIONS('CONNECT','DATASET')
```

Formatting a SYSLOGR Trace

Format the trace with an IPCS CTRACE COMP(SYSLOGR) subcommand. IBM might request that you enter options for SYSLOGR formatting. You can specify options for SYSLOG0R tracing on the OPTIONS parameter of the CTRACE command. The options include:

COMPERR
Displays internal system logger component errors.

CONNECT
Displays list structure connections, disconnections, rebuild and event exit processing.

DATASET
Displays log stream data set allocation and management.

INVENTRY
Displays log stream and structure definition and deletion processing as well as LOGR policy processing.

LOCBUFF
Displays system logger local buffer management.

LOGSTRM
Displays log stream functional request processing.

MISC
Displays system logger internal miscellaneous services.

RECOVERY
Displays system logger component recovery, detecting abnormal conditions during processing.

RQE(*request_address***)**
Specify an 8-byte hexadecimal RQE address. Displays the specified RQE control block.

SERIAL
Displays system logger serialization services.

STACK(*request_address***)**
Specify an 8-byte hexadecimal stack address. Displays the stack at the specified address.

STORAGE
Displays system logger storage management.

Output from a SYSLOGR Trace

CTRACE COMP(SYSLOGR) Subcommand Output

The following is an example of system logger component trace records formatted with the CTRACE COMP(SYSLOGR) subcommand:

```
COMPONENT TRACE FULL FORMAT
COMP(SYSLOGR)
**** 09/12/1994

CONNECT   03190001  13:03:28.955894   System Logger Services
   C9E7C3E2  C9C7F0F3  40404040  40404040  | IXCSIG03          |
   E2D3C3E3  C5E2E3F1  0100                | SLCTEST1..        |
RECOVERY  07040001  13:03:58.055519   System Logger Services
   C9E7C7C3  F4D9C6C3  840C1000  00000001  | IXGC4RFCd.......  |
   03171D80  0000                          | ......            |
RECOVERY  07040001  13:09:55.907719   System Logger Services
   C9E7C7C3  F4D9C6C3  840C1000  00000001  | IXGC4RFCd.......  |
   031700A0  0000                          | ......            |
COMPERR   01070007  13:30:58.322696   System Logger Services
   E2E8E2F0  F0F0F0F1  C9E7C7D3  D6C7D94B  | SYS00001IXGLOGR.  |
   E2C3D6E3  E3F34BC1  F0F0F0F0  F0F0F140  | SCOTT3.A0000001   |
   40404040  40404040  40404040  40404040  |                   |
   40404040  0000                          |     ..            |
```

SYSOMVS Component Trace

Before Using This Component Trace

This topic assumes you have read:
- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSOMVS component trace for z/OS UNIX.

Information	For SYSOMVS:
Parmlib member	CTnBPXxx Default member: CTIBPX00 specified in BPXPRM00 member
Default tracing	Yes; minimal; unexpected events
Trace request OPTIONS parameter	In CTnBPXxx and REPLY for TRACE command
Buffer	• Default: 128KB • Range: 16K - 4M • Size set by: CTnBPXxx member • Location: Data space. In the REPLY for the DUMP command, specify DSPNAME=(*asid*.SYSZBPX1) where *asid* is the ASID for z/OS UNIX.
Trace records location	Data space buffer, trace data set
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSOMVS)
Trace format OPTIONS parameter	Yes

Requesting a SYSOMVS Trace

Specify options for requesting a SYSOMVS component trace on a CTnBPXxx parmlib member or on the reply for a TRACE CT command.

You can change options for SYSOMVS tracing while the trace is running.

CTnBPXxx Parmlib Member

The following table indicates the parameters you can specify on a CTnBPXxx parmlib member.

Parameters	**Allowed on CTnBPXxx?**
ON or OFF	Yes
ASID	Yes
JOBNAME	Yes
BUFSIZE	Yes
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

Note: Specify the new buffer size in the BUFSIZE parameter on the CTnBPXxx member being used.

IBM supplies the CTIBPX00 parmlib member, which specifies the z/OS UNIX tracing begun at ipl. The contents of CTIBPX00 are:

```
TRACEOPTS
      ON
      BUFSIZE(128K)
```

The parameters turn on the minimal SYSOMVS tracing. These parameters request the unexpected or important z/OS UNIX System Services events. The trace buffer size is 128KB. This member activates the minimal trace at ipl. In the IBM-supplied BPXPRM00 parmlib member, the CTRACE parameter specifies CTIBPX00 as the default.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	**Allowed on TRACE CT for Trace?**
ON or OFF	One is required
nnnnK or nnnnM	No
COMP	Required
SUB	No
PARM	Yes

Parameters	Allowed on TRACE CT for Writer?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	Yes
JOBNAME	Yes
OPTIONS	Yes
WTR	Yes

You can change options while a SYSOMVS trace is running.

OPTIONS Parameter

The values for the OPTIONS parameter for the CTnBPXxx parmlib member and reply for a TRACE command, in alphabetical order, are:

ALL
Traces all events.

CHARS
Traces character special events.

DEVPTY
Traces pseudoterminal events.

DEVRTY
Traces outboard communication server (OCS) remote terminal events.

FILE
Traces file system events.

IPC
Traces interprocess communication activity for shared memory, message queues and semaphores.

LOCK
Traces locking services events.

PIPE
Traces pipe events.

PROCESS
Traces process events.

PTRACE
Traces PTRACE events.

SIGNAL
Traces signaling events.

STORAGE
Traces storage management events.

SYSCALL
Traces callable service layer events.

Examples of Requesting SYSOMVS Traces

Example 1: CTnBPXxx Member

The member requests DEVPTY, FILE, and SIGNAL options.

```
TRACEOPTS
   ON
   OPTIONS('DEVPTY','FILE','SIGNAL')
```

Example 2: TRACE Command

The example requests a trace of DEVPTY and FILE trace events.

```
TRACE CT,ON,COMP=SYSOMVS
* 18 ITT006A ...
REPLY 18,OPTIONS(DEVPTY,FILE),END
```

Formatting a SYSOMVS Trace

Format the trace with an IPCS CTRACE COMP(SYSOMVS) subcommand. The OPTIONS parameter specifies the options that select trace records to be formatted. Your formatting options depend to a great extent on the tracing options you requested. Use the options to narrow down the records displayed so that you can more easily locate any errors. If the CTRACE subcommand specifies no options, IPCS displays all the trace records.

ALL
: Formats all events.

CHARS
: Formats character special events.

DEVPTY
: Formats pseudoterminal events.

DEVRTY
: Formats OCS remote terminal events.

FILE
: Formats file system events.

IPC
: Formats events for shared memory, message queues and semaphores.

LOCK
: Formats locking services events.

PIPE
: Formats pipe events.

PROCESS
: Formats process events.

PTRACE
: Formats PTRACE events.

SCCOUNTS
: Counts the number of syscalls that occur in the trace. Also counts the number of function codes that occur in a trace. The outputs are displayed in tables. Formatting is suppressed. The function codes refer to the types of messages that are crossing between systems in a sysplexed environment. In a non-sysplex dump, the functions code table will be empty. You could run an

application while collecting CTRACE data, and then use this option to determine the frequency of syscalls and function codes being made by the application.

SEARCH
Starting at the specified offset, searches trace entries for a specific value and displays the matches.

A CLIST called BPXMSCER is provided to allow the search to be performed against specific entity ids that will identify syscall exits that have failed.

SIGNAL
Formats signaling events.

STORAGE
Formats storage management events.

SYSCALL
Formats callable service layer events.

SYSID(nnn)
Formats sysplex system events.

When this is requested by the user, only those trace records that contain a sysplex system id will be formatted and displayed. (nnn) is the sysplex number or name of the system in the sysplex whose records will be displayed. See "Example of CTRACE DISPLAY PARAMETERS Panel" for an example of a CTRACE DISPLAY PARAMETERS panel and the SYSID option on that panel.

XCF
Formats XCF events.

Example of CTRACE DISPLAY PARAMETERS Panel

The CTRACE DISPLAY PARAMETERS panel has the following format:

```
--------- CTRACE DISPLAY PARAMETERS --------- Enter option
COMMAND ===>

System          ===>  (System name or blank
Component       ===>  SYSOMVS (Component name (required)
Subnames        ===>

GMT/LOCAL       ===>  G      (G or L, GMT is default)
Start/time      ===>         (mm/dd/yy,hh:mm:ss:dddddd or
Stop time       ===>          mm/dd/yy,hh.mm:ss.dddddd)
Limit           ===>  0       Exception ===>
Report type     ===>  SHORT  (SHort, SUmmary, Full, Tally)
User exit       ===>         (Exit program name)
Override source ===>
Options         ===>  SYSID(1)

To enter/verify required values, type any character
Entry IDs ===> Jobnames ===> ASIDs ===> OPTIONS ===> SUBS===>

CTRACE COMP(SYSOMVS) SHORT OPTIONS((SYSID(1)))

ENTER = update CTRACE definition.  END/PF3 = return to previous panel.
S = start CTRACE.  R = reset all fields.
```

When SYSID is specified, only those trace records that contain a sysplex system id will be formatted and displayed. If SYSID is not specified, data from all the systems will be displayed.

Examples of Subcommands to Format a SYSOMVS Trace

Example 1: CTRACE Subcommand Requesting SEARCH Option

The example requests the SEARCH option to search every CTRACE entry, starting at the offset specified by *offset*, for the value specified by *value*.

```
CTRACE COMP(SYSOMVS) FULL OPTIONS((SEARCH(x'offset',x'value'))
```

Example 2: CTRACE Subcommand Requesting SCCOUNTS Option

The example requests the SCCOUNTS option to count the number of syscalls from within the trace.

```
CTRACE COMP(SYSOMVS) FULL OPTIONS((SCCOUNTS))
```

Output from a SYSOMVS Trace

CTRACE COMP(SYSOMVS) FULL Subcommand Output

The following is an example of SYSOMVS component trace records formatted with the CTRACE COMP(SYSOMVS) subcommand.

```
COMPONENT TRACE FULL FORMAT
COMP(SYSOMVS)
**** 05/25/1999

SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------
SY1       XCF       0D890407  18:14:14.551107  XCF BUFFER I/O TRACE

    ASID..0025      USERID....WELLIE1   STACK@....2566DF18
    TCB...009E04A0  EUID......0000000B  SYSCALL...00000036
   +0000  E2C5D5C4  80180101  02000001  000A0002  | SEND............ |
   +0010  B2DBC852  285F5AC7  7BA70500  403E3000  | ..H..¬!G#x.. ... |
   +0020  01FF0006  00008178  C6000000            | ......a.F...     |
SY1       XCF       0D6F0401  18:14:14.551325  NXMSO-->XCF MESSAGE OUT

    ASID..0025      USERID....WELLIE1   STACK@....2566DF18
    TCB...009E04A0  EUID......0000000B  SYSCALL...00000036
   +0000  E2E8D5C3  80A001D3  0A010014  01170BD4  | SYNC...L.......M |
   +0010  0013C000  02000001  000A0002  00004C4B  | ..{...........<. |
   +0020  7BA70500  000080E0  00000000  00000000  | #x.....\........ |
   +0030  0D6F0000  00030025  009E04A0  0000000B  | .?.............. |
   +0040  00000000  2538E980  01000000            | ......Z.....     |
SY1       XCF       0D690402  18:14:14.554457  NXMSG-->XCF MESSAGE SRB EXIT

    ASID..000E      USERID....OMVS      STACK@....25385F28
    TCB...00000000  EUID......00000000  SYSCALL...00000000
   +0000  D9C5E2D7  B4600101  009D6C68  00000080  | RESP.-....%..... |
   +0010  00030000  0A010014  01170BD4  0013402C  | ...........M.. . |
   +0020  02000001  000A0002  00000118  01FF0006  | ................ |
   +0030  00300098  24C02910  00008000  00000000  | ...q.{.......... |
   +0040  00000000  00000000  00000000            | ............     |
SY1       XCF       0D6F0401  18:14:14.554513  NXMSO-->XCF MESSAGE OUT

    ASID..0025      USERID....WELLIE1   STACK@....2566DF18
    TCB...009E04A0  EUID......0000000B  SYSCALL...00000036
   +0000  C6D9C5C5  10000100  00000000  00000000  | FREE............ |
   +0010  00000000  00000000  00000000  00000000  | ................ |
   +0020  00000000  00000000  00000000  00000000  | ................ |
   +0030  0D6F0000  00000000  00000000  00000000  | .?.............. |
   +0040  00000000  2538E980  00000000            | ......Z.....     |
```

Figure 11-2. SYSOMVS Component Trace Formatted with CTRACE COMP(SYSOMVS) (Part 1 of 2)

```
SY2       XCF       0D690402  18:14:14.553698  NXMSG-->XCF MESSAGE SRB EXIT

    ASID..000E      USERID....OMVS      STACK@....25389F28
    TCB...00000000  EUID......00000000  SYSCALL...00000000
   +0000  D9C5C3E5  D4600201  009E04A0  002580E0  | RECVM-.........\ |
   +0010  00030000  0A010014  01170BD4  0013C000  | ...........M..{. |
   +0020  01000001  000A0001  00008178  01FF0006  | ..........a..... |
   +0030  40060098  80000009  00000000  00000000  |  ..q............ |
   +0040  00000000  00000000  4F4F4F4F            | ........||||     |
SY2       XCF       0D6D0403  18:14:14.553715  NXWRK-->XCF WORKER TASK TR.

    ASID..0025      USERID....OMVS      STACK@....25CF0000
    TCB...009E04A0  EUID......0000000B
    FCODE.0003      SYSNAME...SY1
   +0000  E6D6D9D2  3000022C  009D6C68  0A010014  | WORK......%..... |
   +0010  01170BD4  0013C02C  01000001  000A0001  | ...M..{......... |
   +0020  01FF0006  40060098  000080E0  009E04A0  | .... ..q...\.... |
   +0030  00250003  00000000  00000000  00000000  | ................ |
   +0040  00000000                                | ....             |
SY2       XCF       0D890407  18:14:14.553881  XCF BUFFER I/O TRACE

    ASID..0025      USERID....OMVS      STACK@....25CF0000
    TCB...009E04A0  EUID......0000000B
    FCODE.0003      SYSNAME...SY1
   +0000  E2C5D5C4  80180101  01000001  000A0001  | SEND............ |
   +0010  B2DBC852  29114142  7F636AD8  401FB000  | ..H....."..Q ... |
   +0020  01FF0006  00000118  C6000000            | ........F...     |
SY2       XCF       0D6F0401  18:14:14.554039  NXMSO-->XCF MESSAGE OUT

    ASID..0025      USERID....OMVS      STACK@....25CF0000
    TCB...009E04A0  EUID......0000000B
    FCODE.0003      SYSNAME...SY1
   +0000  D9C5E2D7  202002D3  0A010014  01170BD4  | RESP...L.......M |
   +0010  0013402C  01000001  000A0001  00002BBC  | .. ............. |
   +0020  7F636AD8  00000080  00000000  00000000  | "..Q............ |
   +0030  0D6F0000  00030000  009D6C68  00000000  | .?........%..... |
   +0040  253A4088  00000000  00000000            | .. h........     |
```

Figure 11-2. SYSOMVS Component Trace Formatted with CTRACE COMP(SYSOMVS) (Part 2 of 2)

SY1 Trace Flow: Figure 11-3 on page 11-89 and Figure 11-4 on page 11-89 contain the SY1 trace information found in Figure 11-2 on page 11-87.

Figure 11-3 on page 11-89 describes the CTRACE entries generated by the BPXNXMSO processing on the client side. The ASID / TCB highlighted describe the client making the request.

The most important information is the Unique Request-ID (as noted with an asterisk (*)). This is used to track a request from the client to through the server, and back again.

Two separate trace entries are provided: One states that a message has been entered into a block of messages, and the other states that the block has been written. The buffer address (as noted with an @) is used to cross reference these two trace entries.

```
SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------
SY1       XCF       0D890407  18:14:14.551107  XCF BUFFER I/O TRACE

   #ASID..0025      USERID....WELLIE1   STACK@....2566DF18
   #TCB...009E04A0  EUID......0000000B  SYSCALL...00000036
   +0000  E2C5D5C4  80180101  02000001  000A0002  | SEND............ |
   +0010  B2DBC852  285F5AC7 @7BA70500  403E3000  | ..H..¬!G#x.. ... |
   +0020  01FF0006  00008178  C6000000            | ......a.F...     |

SY1       XCF       0D6F0401  18:14:14.551325  NXMSO-->XCF MESSAGE OUT

   #ASID..0025      USERID....WELLIE1   STACK@....2566DF18
   #TCB...009E04A0  EUID......0000000B  SYSCALL...00000036
   +0000  E2E8D5C3  80A001D3 !0A010014 *01170BD4  | SYNC...L.......M |
   +0010 $0013C000  02000001  000A0002  00004C4B  | ..{...........<. |
   +0020 @7BA70500  000080E0  00000000  00000000  | #x.....\........ |
   +0030  0D6F0000  00030025  009E04A0  0000000B  | .?.............. |
   +0040  00000000  2538E980  01000000            | ......Z.....     |

# - ASID / TCB of requester    @ - Buffer address containing request
$ - Block #, Index into NXRQ   ! - HFS function being requested
* - Unique Request-ID = System number w/ 3byte seq#
```

Figure 11-3. SY1 Trace Flow: Part 1

Figure 11-4 describes the response arriving on the client system. First the XCF SRB (BPXNXMSG) processes the incoming response to cause the client task to be activated. And then the target task (no longer remapped) wakes up, and, in this example, explicitly frees the resources that were allocated to it as part of the XCF message processing.

```
SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------
SY1       XCF       0D690402  18:14:14.554457  NXMSG-->XCF MESSAGE SRB EXIT

    ASID..000E      USERID....OMVS      STACK@....25385F28
    TCB...00000000  EUID......00000000  SYSCALL...00000000
   +0000  D9C5E2D7  B4600101  009D6C68  00000080  | RESP.-....%..... |
   +0010  00030000 !0A010014 *01170BD4 $0013402C  | ...........M.. . |
   +0020  02000001  000A0002  00000118  01FF0006  | ................ |
   +0030  00300098  24C02910  00008000  00000000  | ...q.{.......... |
   +0040  00000000  00000000  00000000            | ............     |

SY1       XCF       0D6F0401  18:14:14.554513  NXMSO-->XCF MESSAGE OUT

   #ASID..0025      USERID....WELLIE1   STACK@....2566DF18
   #TCB...009E04A0  EUID......0000000B  SYSCALL...00000036
   +0000  C6D9C5C5  10000100  00000000  00000000  | FREE............ |
   +0010  00000000  00000000  00000000  00000000  | ................ |
   +0020  00000000  00000000  00000000  00000000  | ................ |
   +0030  0D6F0000  00000000  00000000  00000000  | .?.............. |
   +0040  00000000  2538E980  00000000            | ......Z.....     |

# - ASID / TCB of requester
$ - Block #, Index into NXRQ   ! - HFS function being requested
* - Unique Request-ID
```

Figure 11-4. SY1 Trace Flow: Part 2

SY2 Trace Flow: Figure 11-5 and Figure 11-6 on page 11-91 contain the SY2 trace information found in Figure 11-2 on page 11-87.

Figure 11-5 describes the server side XCF SRB processing by first queuing the request (BPXNXMSG), and then having a worker task pick up that piece of work for subsequent processing (BPXNXWRK).

As noted with an *, the Request-ID is used to cross reference the individual trace entries.

When a SYSNAME field is included in a trace entry, the ASID / TCB information actually describes the client side requester information. The SYSNAME field describes the originating system. The highlighted field with an & is the TCB address of the worker task resident in the server system.

```
SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------
SY2       XCF       0D690402  18:14:14.553698  NXMSG-->XCF MESSAGE SRB EXIT

    ASID..000E      USERID....OMVS      STACK@....25389F28
    TCB...00000000  EUID......00000000  SYSCALL...00000000
   +0000  D9C5C3E5  D4600201  009E04A0  002580E0  | RECVM-.........\ |
   +0010  00030000 !0A010014 *01170BD4 $0013C000  | ...........M..{. |
   +0020  01000001  000A0001  00008178  01FF0006  | ..........a..... |
   +0030  40060098  80000009  00000000  00000000  |  ..q............ |
   +0040  00000000  00000000  4F4F4F4F            | ........||||     |

SY2       XCF       0D6D0403  18:14:14.553715  NXWRK-->XCF WORKER TASK TRACE

   #ASID..0025      USERID....OMVS      STACK@....25CF0000
   #TCB...009E04A0  EUID......0000000B
    FCODE.0003      SYSNAME...SY1
   +0000  E6D6D9D2  3000022C &009D6C68 !0A010014  | WORK......%..... |
   +0010 *01170BD4 $0013C02C  01000001  000A0001  | ...M..{......... |
   +0020  01FF0006  40060098  000080E0  009E04A0  | .... ..q...\.... |
   +0030  00250003  00000000  00000000  00000000  | ................ |
   +0040  00000000                                | ....             |

# - ASID / TCB of requester    & - Real OMVS resident worker TCB
$ - Block #, Index into NXRQ   ! - HFS function being requested
* - Unique Request-ID          % - Indicates ASID/TCB remapped to requester
```

Figure 11-5. SY2 Trace Flow: Part 1

Figure 11-6 on page 11-91 describes the response arriving on the client system. First the XCF SRB (BPXNXMSG) processes the incoming response to cause the client task to be activated. And then the target task (no longer remapped) wakes up, and in this case explicitly frees the resources that were allocated to it as part of the XCF message processing.

```
SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------
SY2       XCF       0D890407  18:14:14.553881  XCF BUFFER I/O TRACE

   #ASID..0025      USERID....OMVS      STACK@....25CF0000
   #TCB...009E04A0  EUID......0000000B
    FCODE.0003     %SYSNAME...SY1
   +0000  E2C5D5C4  80180101  01000001  000A0001  | SEND............ |
   +0010  B2DBC852  29114142 @7F636AD8  401FB000  | ..H....."..Q ... |
   +0020  01FF0006  00000118  C6000000            | ........F...     |

SY2       XCF       0D6F0401  18:14:14.554039  NXMSO-->XCF MESSAGE OUT

   #ASID..0025      USERID....OMVS      STACK@....25CF0000
   #TCB...009E04A0  EUID......0000000B
    FCODE.0003     %SYSNAME...SY1
   +0000  D9C5E2D7  202002D3 !0A010014 *01170BD4  | RESP...L.......M |
   +0010 $0013402C  01000001  000A0001  00002BBC  | .. ............. |
   +0020 @7F636AD8  00000080  00000000  00000000  | "..Q............ |
   +0030  0D6F0000  00030000  009D6C68  00000000  | .?........%..... |
   +0040  253A4088  00000000  00000000            | .. h........     |

# - ASID / TCB of requester    @ - Buffer address containing request
$ - Block #, Index into NXRQ   ! - HFS function being requested
* - Unique Request-ID          % - Indicates ASID/TCB remapped to requester
```

Figure 11-6. SY2 Trace Flow: Part 2

CTRACE COMP(SYSOMVS) SUMMARY Subcommand Output

The following is an example of SYSOMVS component trace records formatted with the CTRACE COMP(SYSOMVS) SUMMARY subcommand.

```
COMPONENT TRACE SHORT FORMAT
COMP(SYSOMVS)
**** 06/17/92

MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
--------  --------  ---------------  -----------

SYSCALL   0F080002  22:02:42.314264  STANDARD SYSCALL EXIT TRACE
SYSCALL   0F080001  22:02:42.821156  STANDARD SYSCALL ENTRY TRACE
PROCESS   0B080007  22:02:42.821219  PROCESS LATCH OBTAIN
PROCESS   0B08000A  22:02:42.821256  PROCESS LATCH-ON THE WAY OUT
FILE      05700103  22:02:42.821324  TRACE CALL TO VN_RDWR
CHARS     05A90503  22:02:42.821398  TRACE CHARSPEC CALL TO DEVICE DR
PROCESS   0B080008  22:02:42.821452  PROCESS LATCH RELEASE REQUEST
PROCESS   0B08000A  22:02:42.821472  PROCESS LATCH-ON THE WAY OUT
DEVPTY    0223E005  22:02:42.821530  MASTER READ BEGIN
DEVPTY    0223E008  22:02:42.821566  MASTER READ END
PROCESS   0B080007  22:02:42.822182  PROCESS LATCH OBTAIN
PROCESS   0B08000A  22:02:42.822206  PROCESS LATCH-ON THE WAY OUT
CHARS     05A90603  22:02:42.822253  TRACE DEVICE DRIVER READ RETURN
FILE      05700203  22:02:42.822506  TRACE RETURN FROM VN_RDWR
```

Output from a SYSOMVS Trace Using the SCCOUNTS Option

The output from a SYSOMVS trace using the SCCOUNTS option has 2 formats, shown in Figure 11-7 on page 11-92 and Figure 11-8 on page 11-92.

```
SYSCALL#     SYSCALL NAME      COUNT      FREQUENCY/SEC
--------     ------------      -----      -------------
    F        BPX1CHO            5000           nnn
   2F        BPX1STA             150           nnn
```

Figure 11-7. SCCOUNT Function Displaying SYSCALL Frequency

Figure 11-7 is sorted by frequency, with the highest values appearing at the top of the list. SYSCALL# is the hexadecimal number of the syscall. FREQUENCY/SEC is the number of times the syscall has been invoked within the interval.

```
 FuncCode    FuncCode Name     Count      Functions/Sec
 --------    ------------      -----      -------------
 00001001    MntCatchup         76           0.0593
 00001010    GetPathName        60           0.0468
 00000012    UnQuiesce          38           0.0296
```

Figure 11-8. SCCOUNT Function Displaying Function Code Frequency

Figure 11-8 is sorted by frequency, with the highest values appearing at the top of the list. FuncCode is the hexadecimal number of the function code. Functions/Sec is the number of times the function code has been invoked within the interval.

SYSOPS Component Trace

Before Using This Component Trace

This topic assumes you have read:

- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSOPS component trace for the operations services component (OPS).

Information	For SYSOPS:
Parmlib member	CTnOPSxx Default member: CTIOPS00 specified in CONSOLxx member
Default tracing	Yes; minimal; unexpected events
Trace request OPTIONS parameter	In CTnOPSxx or REPLY for TRACE command
Buffer	• Default: 64KB • Range: 64KB - 16MB • Size set by: CTnOPSxx member or REPLY for TRACE CT command • Change size after IPL: Yes, while a trace is running • Location: Console address space (private)
Trace records location	Address-space buffer, trace data set
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSOPS)
Trace format OPTIONS Parameter	Yes

Note: To get a complete dump for OPS, request also the NUC, CSA, and SQA.

Requesting a SYSOPS Trace

Specify options for requesting a SYSOPS component trace in a CTnOPSxx parmlib member or in the reply for a TRACE CT command.

CTnOPSxx Parmlib Member

The following table indicates the parameters you can specify on a CTnOPSxx parmlib member.

Parameters	Allowed on CTnOPSxx?
ON or OFF	Yes
ASID	Yes
JOBNAME	Yes
BUFSIZE	Yes
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

IBM supplies the CTIOPS00 parmlib member, which defines the component trace to the system and establishes a trace buffer of 64KB. The contents of CTIOPS00 are:

```
TRACEOPTS
      ON
      BUFSIZE(64K)
```

IBM does not supply a sample CONSOL00 parmlib member. Create a CONSOLxx parmlib member and specify CON=xx in the IEASYSxx parmlib member. Specify CTIOPS00 as the default on the CTRACE parameter of the INIT statement of CONSOLxx.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON, nnnnK, nnnnM, or OFF	One is required
COMP	Required
SUB	No
PARM	Yes

Parameters	Allowed on TRACE CT for Write?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	Yes

Parameters	Allowed on REPLY for Trace?
JOBNAME	Yes
OPTIONS	Yes
WTR	Yes

OPTIONS Parameter

The values for the OPTIONS parameter for the CTnOPSxx parmlib member and reply for a TRACE command, in alphabetical order, are:

MESSAGES
This option includes the WTO and MSGDLVRY options.

MSG=*msgid*
Trace processing of a specific message. It REQUIRES either one of the following OPTIONS: MESSAGES, WTO or MSGDLVRY. *msgid* is 1-10 alphanumeric characters in length indicating the message id that will be traced.

MSGDLVRY
Traces events for WQE processing, MCS console message queueing, and extended MCS console message processing.

SYSPLEX
Traces events for XCF signalling, sysplex serialization services, sysplex clean-up processing, and the manipulation of various queues.

WTO
Traces the effects of MPFLSTxx, the user exits, and the SSI on message content and attributes.

These additional options increase the number of trace records the system collects and can slow system performance. Each time you change the trace options, you must respecify any options you want to keep in effect from the last trace.

Note: Before you use the MESSAGES, WTO, or MSGDLVRY options, you should do the following:

- Increase the buffer size
- Start and connect the external writer.

This is especially important if you are starting tracing at IPL This might not be necessary if you are tracing a message ID since you would only be cutting records for the particular message.

Examples of Requesting SYSOPS Traces

Example 1: Activating Trace Options While System is Running

Create parmlib member CTIOPS01, specifying the following parameters. Assume that procedure OPSWTR is in SYS1.PROCLIB.

```
TRACEOPTS
      WTRSTART(OPSWTR)
      ON
      WTR(OPSWTR)
      OPTIONS('MESSAGES','MSG=IEE136I')
       ASID(1,2,3)
      JOBNAME(PAYROLL)
      BUFSIZE(2M)
```

Example 2: Specifying Trace Options on a REPLY Command

The example requests the same trace as Example 1, but specifies all options on the REPLY.

```
trace ct,wtrstart=opswtr
trace ct,2m,comp=sysops
```

When the system prompts you for the trace options, enter the following command, replacing *id* with the reply identifier:

```
reply 27,wtr=opswtr,options=(messages,msg=IEE136I),asid=(1,2,3),
jobname=(payroll),end
```

Example 3: CTnOPSxx Member Used at IPL

The member requests the SYSPLEX option, doubles the default buffer size, and limits the tracing to ASID 1 and JOBNAME JOB1.

```
TRACEOPTS
      ON
      OPTIONS('SYSPLEX')
      BUFSIZE(128K)
      ASID(1)
      JOBNAME(JOB1)
```

Formatting a SYSOPS Trace

Format the trace with an IPCS CTRACE COMP(SYSOPS) subcommand. The OPTIONS parameter specifies the options that select trace records to be formatted. Your formatting options depend to a great extent on the tracing options you requested. Use the options to narrow down the records displayed so that you can more easily locate any errors. If the CTRACE subcommand specifies no options, IPCS displays all the trace records. The options for formatting a SYSOPS trace are:

MESSAGES
This option includes the WTO and MSGDLVRY options.

MSG=*msgid*
Trace processing of a specific message. It REQUIRES either one of the following OPTIONS: MESSAGES, WTO or MSGDLVRY. *msgid* is 1-10 alphanumeric characters in length indicating the message id that will be traced.

MSGDLVRY
Traces events for WQE processing, MCS console message queueing, and extended MCS console message processing.

SYSPLEX
Traces events for XCF signalling, sysplex serialization services, sysplex clean-up processing, and the manipulation of various queues.

WTO
Traces the effects of MPFLSTxx, the user exits, and the SSI on message content and attributes.

Output from a SYSOPS Trace

CTRACE COMP(SYSOPS) SHORT Subcommand Output

The following is an example of OPS component trace records formatted with the CTRACE COMP(SYSOPS) SHORT subcommand:

```
COMPONENT TRACE SHORT FORMAT
COMP(SYSOPS)
**** 05/20/93

MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
--------  --------  ---------------  -----------

INITMSG   00000001  14:41:06.918883  Message prior to MPF processing
POSTMPF   00000002  14:41:06.919198  Message after MPF processing
POSTEXIT  00000003  14:41:06.919595  Post Message Exit
POSTSSI   00000004  14:41:06.920118  Post Subsystem Interface
INITMSG   00000001  14:41:06.930088  Message prior to MPF processing
POSTMPF   00000002  14:41:06.930405  Message after MPF processing
POSTEXIT  00000003  14:41:06.930803  Post Message Exit
POSTSSI   00000004  14:41:06.931267  Post Subsystem Interface
INITMSG   00000001  14:41:06.931637  Message prior to MPF processing
POSTMPF   00000002  14:41:06.931934  Message after MPF processing
```

CTRACE COMP(SYSOPS) FULL Subcommand Output

The following is an example of OPS component trace records formatted with the CTRACE COMP(SYSOPS) FULL subcommand:

```
COMPONENT TRACE FULL FORMAT
COMP(SYSOPS)
**** 05/20/93

MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
--------  --------  ---------------  -----------

INITMSG   00000001  14:41:06.918883  Message prior to MPF processing
     TR_GROUP WTO               Write to Operator Services
     HOMEASID 0001     HOMEJOBN *MASTER*
     REQSASID 0001     REQSJOBN UNKNOWN
     CPU_ADDR 0001
     KEY..... 0001
   +0000  LKP...... 00000000  TXTLN.... 005F      RTCT..... 0000
   +000A  USE...... 0000      PAD...... 40        TS....... 10.41.06
   +00D1  TS2...... .91       PAD1..... 40        JOBNM....
   +001E  PAD2..... 40        TXT......  ITT038I ALL OF THE TRANSACTIONS REQU
   +0044            ESTED VIA THE TRACE CT COMMAND WERE
   +009E  TXTL..... 40        PAD3..... 00        XA....... 01
   +00A1  ASID..... 0001      AVAIL.... 50        TCB...... 008D57C0
   +00A8  SYSID.... 03        SEQN..... 00049D    MCSF1.... C0
   +00AD  MCSF2.... 10        MSGT1.... 00        ROUT1.... 00
   +00B1  ROUT2.... 00        CHAR1.... 00        FLG3..... 00
   +00B4  UCMID.... 0A        FLG1..... 04        RPYID.... 0000
   +00B8  DC1...... 08        DC2...... 00        RSV26.... 0000
   +00BC  JSTCB.... 008D57C0  VRSN..... 05        MFLG1.... 00
   +00C2  MCSE1.... 42        MCSE2.... 00        SYSNM.... SYS2
   +00CC  DATE..... 93140     RFB1..... 00        RFB2..... 00
   +00D6  RFB3..... 00        SUPB..... 90        ML1...... 00
   +00D9  ML2...... 00        LENG..... 0160      DSQN..... 00000000
   +00E0  ERC1..... 00        ERC2..... 00        ERC3..... 00
   +00E3  ERC4..... 00        ERC5..... 00        ERC6..... 00
   +00E6  ERC7..... 00        ERC8..... 00        ERC9..... 00
   +00E9  ERC10.... 00        ERC11.... 00        ERC12.... 00
   +00EC  ERC13.... 00        ERC14.... 00        ERC15.... 00
   +00EF  ERC16.... 00        KEY...... 40404040  40404040
   +00F8  TOKN..... 00000000  CNID..... 0000000A  OJBID.... 40404040
   .
   .
   .
```

SYSRRS Component Trace

Before Using This Component Trace

This topic assumes you have read:
- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSRRS component trace for RRS.

Information	For SYSRRS:
Parmlib member	CTnRRSxx No default member

Information	For SYSRRS:
Default tracing	Yes; minimal; unexpected events
Trace request OPTIONS parameter	In CTnRRSxx and REPLY for TRACE command
Buffer	• Default: 1MB • Range: 1MB - 2045MB • Size set by: CTnRRSxx member or REPLY for TRACE CT command • Change size after IPL: Yes, when restarting a trace after stopping it • Location: Data space and component address space. In the REPLY for the DUMP command, specify DSPNAME=('RRS'.ATRTRACE) and SDATA=RGN.
Trace records location	Data space buffer, trace data set, trace buffers in the RRS address space
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSRRS)
Trace format OPTIONS parameter	Yes

Requesting a SYSRRS Trace

Specify options for requesting a SYSRRS component trace on a CTnRRSxx parmlib member or on the reply for a TRACE CT command.

To change options or the buffer size, you have to stop the trace and restart it with the new options, buffer size, or both.

CTnRRSxx Parmlib Member

The following table indicates the parameters you can specify on a CTnRRSxx parmlib member.

Parameters	Allowed on CTnRRSxx?
ON or OFF	Yes
ASID	Yes
JOBNAME	Yes
BUFSIZE	Yes
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

A CTnRRSxx parmlib member is optional. If not specified, the SYSRRS component trace runs a minimal trace beginning when the RRS component is started and ending when the component is stopped.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on a TRACE CT command and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON, nnnnK, nnnnM, or OFF	One is required
COMP	Required
SUB	No
PARM	Yes

Parameters	Allowed on TRACE CT for Writer?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	Yes
JOBNAME	Yes
OPTIONS	Yes
WTR	Yes

OPTIONS Parameter

The OPTIONS parameter for the CTnRRSxx parmlib member or reply for a TRACE command contains keyword subparameters. These subparameters allow you to control the information that RRS component trace collects. The first subparameter, EVENTS, specifies the events to be traced; the other subparameters act as filters to screen the events with up to three checks. An event must pass all checks for component trace to generate a trace record. The order for the checks is:

1. That the event is specified.
2. That the event matches, in an OR check, one of the following filters:
   - The address space ID (ASID) in the REPLY command or CTnRRSxx TRACEOPTS statement
   - The job name (JOBNAME) in the REPLY command or CTnRRSxx TRACEOPTS statement
   - The user ID (USERID) in the OPTIONS parameter
   - The logical work unit ID (LUWID) in the OPTIONS parameter
3. That the event matches a resource manager name (RMNAME) in the OPTIONS parameter.

```
OPTIONS=([EVENTS(event[,event]...)]
         [USERID(userid[,userid]...)]
         [RMNAME(rmname[,rmname]...)]
         [LUWID((luwid)[,(luwid)]...)])
         [EID((eid)[,(eid)]...)
```

Note: In the REPLY to the TRACE CT command, separate the options by one or more blanks.

EVENTS(*event***[,***event***]...)**
Indicates the events to be traced. **An EVENTS parameter is required if any other options are specified.** The events, in alphabetical order, are:

ALL
Traces all events. Component trace ignores other events, if specified.

CONTEXT
Traces calls to context services.

EXITS
Traces events related to running the RRS exit routines provided by the resource managers.

FLOW
Traces entry into and exit from RRS entry points.

LOGGING
Traces events related to logging data by RRS.

RESTART
Traces events related to RRS initialization and restart.

RRSAPI
Traces events related to the application programming interface, which consists of calls to the Application_Commit_UR service and the Application_Backout_UR service.

STATECHG
Traces events involving changes in the state of units of recovery (URs).

URSERVS
Traces general events related to services for a UR.

USERID(*userid***[,***userid***]...)**
Specifies 1 to 16 user IDs as filters for specified events. The system traces only events relating to the user IDs.

RMNAME(*rmname***[,***rmname***]...)**
Specifies 1 to 16 resource manager names as filters for specified events. For trace events sensitive to the resource manager name, the system traces only events relating to the resource managers.

LUWID((*luwid***)[,(***luwid***)]...)**
Specifies 1 to 16 logical unit of work identifiers (LUWIDs) as filters for specified events. The system traces only events relating to the LUWIDs. Each *luwid* consists of:

```
netid.luname[,instnum][,seqnum]
```

Component trace ignores leading and trailing blanks.

netid.luname
Specifies the network ID and the local logical unit name. These portions of the LUWID are required.

You can use an asterisk (*) as a wildcard character as:

- The last character in the *netid*, the *luname*, or both
- The only character in the either the *netid* or the *luname*, but not as the only character in both

instnum
Specifies the instance number as a 1 - 12 hexadecimal integer. You can omit leading zeros.

seqnum
Specifies the sequence number as a 1 - 4 hexadecimal integer. You can omit leading zeros.

Examples of LUWIDs are:

```
(A.B,5,1)
(A.B,5)
(A.B,,1)
(A.B)
(A.*,5,1)
(A*.B*)
```

EID((*eid*)[,(*eid*)]...)
Specifies 1 to 16 Enterprise identifiers (EIDs) as filters for specified events. The system traces only events relating to the EIDs. Each *eid* consists of:

```
[tid][,gtid]
```

You can omit leading zeros. Component trace ignores leading and trailing blanks.

tid
Specifies the 4-byte hexadecimal transaction identifier (TID).

gtid
Specifies the 8-40 byte hexadecimal global transaction identifier (GTID).

You can obtain the EID for a UR by using the RRS ISPF panels to browse the RRS log streams. The Retrieve_Work_Identifier service can also return an EID.

Examples of EIDs are:

```
(1,C)
(,C)
(1)
```

Examples of Requesting SYSRRS Traces

Example 1: CTnRRSxx Member

The member requests context services events filtered by the user ID JONES and requests a 1024KB buffer.

```
TRACEOPTS
   ON
   OPTIONS(EVENTS(CONTEXT) USERID(JONES))
   BUFSIZE(1024K)
```

Example 2: TRACE Command

The example requests the same trace as Example 1.

```
trace ct,off,comp=sysrrs
trace ct,1024K,comp=sysrrs
* 17 ITT006A ...
reply 17,options=(events(context) userid(jones)),end
```

Formatting a SYSRRS Trace

Format the trace with an IPCS CTRACE COMP(SYSRRS) subcommand. Its OPTIONS parameter specifies the options that select trace records to be formatted. Your formatting options depend to a great extent on the tracing options you requested. Use the formatting options to narrow down the records displayed so that you can more easily locate any errors. If you specify no options on the CTRACE subcommand, IPCS displays all the trace records.

You can specify one or more OPTIONS subparameters. If you specify no OPTIONS subparameters, all trace records are formatted. A trace record must match all specified OPTIONS subparameters to be formatted.

```
OPTIONS=((option[,option]...))

option is one of the following:

        LUWID(luwid)
        EID(eid)
        RMNAME(rmname)
        URID(urid)
        USERID(userid)
```

LUWID(*luwid*)
Specifies one of the logical unit of work identifiers (LUWIDs) specified when the trace was generated.

EID(*eid*)
Specifies one of the Enterprise identifiers (EIDs) specified when the trace was generated.

Specify *eid* as:

```
tid
tid,gtid
```

Or, if you omitted `tid` when you specified the identifier:

```
*,gtid
```

RMNAME(*rmname*)
Specifies one of the resource manager names specified when the trace was generated.

URID(*urid*)
Specifies a UR identifier. The URID is a 16-byte character string returned to the resource manager by one of the following callable services: Change_Interest_Type, Express_UR_Interest, Retain_Interest, Retrieve_UR_Interest, or Retrieve_UR_Data. The URID is saved in the resource manager log; you can obtain it through an RRS panel.

USERID(*userid*)
Specifies one of the user IDs specified when the trace was generated. Note that USERID does not filter out trace records in which the user ID is blank.

Example of an IPCS CTRACE OPTIONS Parameter

```
OPTIONS=((RMNAME(datamgr),USERID(jjones)))
```

Output from a SYSRRS Trace

Fields that do not contain printable characters are filled with asterisks (*). The value is shown in hexadecimal on a separate line.

Note: RRS provides the same report for the SUMMARY and FULL parameters on the CTRACE subcommand.

CTRACE COMP(SYSRRS) SHORT Subcommand Output

The following is an example of SYSRRS component trace records formatted with the CTRACE COMP(SYSRRS) SHORT subcommand.

```
 COMPONENT TRACE SHORT FORMAT
 COMP(SYSRRS)
 **** 01/20/1997

 SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
 -------   --------  --------  ---------------  -------------

 SY1       COMPERR   00000000  07:53:43.615114  Resource Recovery Services
 SY1       FLOW      0801FFFE  07:53:43.615114  ATRB1PCT EXIT
 SY1       FLOW      0801FFFF  07:53:43.619823  ATRB1PCT ENTRY
 SY1       FLOW      0201FFFF  07:53:43.619867  ATRU1EIN ENTRY
 SY1       CONTEXT   02010002  07:53:43.620027  CTXMEINT call
 SY1       URSERVS   02010001  07:53:43.620699  ATREINT invoked
 SY1       FLOW      0201FFFE  07:53:43.620887  ATRU1EIN EXIT
.
.
.
```

CTRACE COMP(SYSRRS) SUMMARY or FULL Output

```
COMPONENT TRACE FULL FORMAT
COMP(SYSRRS)
**** 01/20/1997

SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------

SY1       COMPERR   00000000  07:53:43.615114  Resource Recovery Services
   FFF0003E  7EF6D000  0001FE6E  00000000  | .0..=6}....>.... |
   000A0000  00000000  00020000  | ................ |
   0101001F  00000000  7F04D000  01000000  | ........".}..... |
   00000000  00000000  0000      | ..............   |
SY1       FLOW      0801FFFE  07:53:43.615114  ATRB1PCT EXIT
     HASID.... 00AA      HJOBNAME. APPL1AS
     SASID.... 00A3      SJOBNAME. RMAS1
     USERID... *         RMNAME...
     URID..... AE18AB4E  7ED2CF90  0000012B  01010000
     TID...... 000000000000                  GTID..... 00000000
               00000000  00000000  00000000
     LUWID.... 00000000  00000000  00000000  0000
       NETNAME.. ********  LUNAME... ********
       INSTNUM.. ******    SEQNUM... **
SY1       FLOW      0801FFFF  07:53:43.619823  ATRB1PCT ENTRY
     HASID.... 00AA      HJOBNAME. APPL1AS
     SASID.... 00A3      SJOBNAME. RMAS1
     USERID... *         RMNAME...
     URID..... AE18AB4E  7ED2CF90  0000012B  01010000
     TID...... 000000000000                  GTID..... 00000000
               00000000  00000000  00000000
     LUWID.... 00000000  00000000  00000000  0000
       NETNAME.. ********  LUNAME... ********
       INSTNUM.. ******    SEQNUM... **
SY1       FLOW      0201FFFF  07:53:43.619867  ATRU1EIN ENTRY
     HASID.... 00AA      HJOBNAME. APPL1AS
     SASID.... 00A3      SJOBNAME. RMAS1
     USERID... *         RMNAME...
     URID..... AE18AB4E  7ED2CF90  0000012B  01010000
     TID...... 000000000000                  GTID..... 00000000
               00000000  00000000  00000000
     LUWID.... 00000000  00000000  00000000  0000
       NETNAME.. ********  LUNAME... ********
       INSTNUM.. ******    SEQNUM... **
.
.
.
```

CTRACE COMP(SYSRRS) TALLY Output

```
COMPONENT TRACE TALLY REPORT
COMP(SYSRRS)
    TRACE ENTRY COUNTS AND AVERAGE INTERVALS (IN MICROSECONDS)

FMTID    COUNT       INTERVAL     MNEMONIC DESCRIBE
-------- ----------- ------------ -------- --------------------------------
00000000           7   23,774,056 COMPERR  Resource Recovery Services
02010001          46    2,398,672 URSERVS  ATREINT invoked
02010002          46    2,398,670 CONTEXT  CTXMEINT call
02018001           6   18,803,430 STATECHG UR State Change
02018008           3   47,008,505 STATECHG UR being created
0201FFFE          46    2,398,673 FLOW     ATRU1EIN EXIT
0201FFFF          46    2,398,670 FLOW     ATRU1EIN ENTRY
02030001           0              URSERVS  ATRDINT invoked
02038009           0              STATECHG UR being destroyed
0203FFFE           0              FLOW     ATRU1DIN EXIT
0203FFFF           0              FLOW     ATRU1DIN ENTRY
02050001           0              URSERVS  ATRPDUE invoked
0205FFFE           0              FLOW     ATRU1PDU EXIT
0205FFFF           0              FLOW     ATRU1PDU ENTRY
.
.
.
```

SYSRSM Component Trace

Before Using This Component Trace

This topic assumes you have read:

- "Planning for Component Tracing" on page 11-3
- "Obtaining a Component Trace" on page 11-10
- "Viewing the Component Trace Data" on page 11-23

The following summarizes information for requesting a SYSRSM component trace for the real storage manager (RSM).

Information	For SYSRSM:
Parmlib member	CTnRSMxx No default member
Default tracing	No
Trace request OPTIONS parameter	In CTnRSMxx or REPLY for TRACE command
Buffer	• Default: 3 buffers of 32 pages • Range: 2 - 7 address-space buffers, 4 - 262,144 pages per buffer 1 - 2047MB for data-space buffers • Size set by: CTnRSMxx member or REPLY for TRACE CT command • Change size after IPL: Yes, when starting a trace • Location: Common service area (CSA) or, if specified in CTnRSMxx, a data space. Message IAR007I provides the data space name.
Trace records location	Address-space buffer, data-space buffer, trace data set

Information	For SYSRSM:
Request of SVC dump	• By DUMP or SLIP command • Through the DMPREC option on the CTnRSMxx parmlib member or on the REPLY for the TRACE CT command when RSM enters recovery processing (default) • Through the DMPOFF option of CTnRSMxx or the TRACE CT reply when SYSRSM tracing is turned off
Trace formatting by IPCS	CTRACE COMP(SYSRSM)
Trace format OPTIONS parameter	None

Requesting a SYSRSM Trace

Specify options for requesting a SYSRSM component trace in a CTnRSMxx parmlib member or in the reply for a TRACE CT command.

CTnRSMxx Parmlib Member

The following table indicates the parameters you can specify on a CTnRSMxx parmlib member.

Parameters	Allowed on CTnRSMxx?
ON or OFF	Yes
ASID	Yes
JOBNAME	Yes. To trace all batch jobs, specify 'INIT' in the list of job names.
BUFSIZE	Yes
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

IBM provides two sample CTIRSMxx parmlib members in SYS1.PARMLIB. These are not default members.

- **CTIRSM01**: Shows how to request tracing of all RSM functions and events using the options DMPOFF, NOCOMASID, and NODMPREC.
- **CTIRSMSP**: Shows how to request address space, job name filtering, and trace request options to limit the tracing.

RSM Trace Data in a Data Space: RSM supports collecting trace data in data spaces. In this case, the system copies trace data from the buffers to a data space. A data space is another way, besides trace data sets, to handle a large number of RSM trace records. Note that the paging activity for the data space can appear in the RSM trace records.

If you suspect that your system has a paging problem, collect the RSM trace records in address-space buffers to keep from losing records while paging. A record can be lost as the system reuses a full data-space buffer. The system gives the number of lost trace records in the first record of the next data-space buffer.

For RSM, the BUFSIZE parameter in the CTnRSMxx parmlib member specifies the size of a data-space buffer; for other components, BUFSIZE specifies the size of the address-space buffer.

Example: CTWRSM05 Parmlib Member

For RSM, use BUFF in the CTnRSMxx OPTIONS parameter to specify the number of address-space buffers and their page sizes. The statements in CTWRSM05 specify 4 address-space buffers that are 64 pages and a data-space buffer of 640KB.

```
TRACEOPTS
   ON
   BUFSIZE(640K)
   OPTIONS('BUFF(4,64)')
```

The name of the data space containing the buffer is provided in message IAR007I. The name has the form `SYSIARnn`. The operator should specify the data space name in the reply for the DUMP command.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON, nnnnK, nnnnM, or OFF	One is required. *nnnnK* and *nnnnM* specify the size of the buffer in a data space.
COMP	Required
SUB	No
PARM	Yes

Parameters	Allowed on TRACE CT for Write?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	Yes
JOBNAME	Yes. To trace all batch jobs, specify ‘INIT’ in the list of job names.
OPTIONS	Yes
WTR	Yes

Automatic Dump: The component requests an SVC dump when the operator stops the trace or when RSM enters recovery processing. To prevent these automatic dumps when the trace is written to a trace data set or when the operator is to request the dump, specify NODMPREC and NODMPOFF in the OPTIONS parameter in the TRACE CT command or the CTnRSMxx parmlib member.

OPTIONS Parameter

The values for the OPTIONS parameter for the CTnRSMxx parmlib member and reply for a TRACE command follow.

If you turn on the SYSRSM trace without specifying any filters or options, the component trace records every RSM function and event in all address spaces and jobs. This trace collects an enormous amount of data and degrades system performance. Use the SYSRSM filters and options to limit the amount of data recorded by the component trace. Specify tracing of specific address spaces, jobs, RSM events, and RSM functions.

The RSM trace options are divided into three groups:
- Special trace options
- RSM function trace options
- RSM event trace options

Special Trace Options: These options set the size of the fixed RSM trace buffers, tell the trace to record common area activity, and tell the system when to dump the trace data. The options are:

BUFF=(x,y)
Specifies the number and size of the SYSRSM trace buffers, which reside in fixed extended common service area (ECSA) storage:

x The number of buffers, from 2 to 7. The default is 3.

y The number of pages per buffer, from 4 to 262,144. The default is 32.

For example, if you specify BUFF=(5,10), the component trace uses 5 fixed trace buffers. Each buffer contains 10 pages. The total amount of fixed storage used is 200 kilobytes.

Note: When choosing the amount of fixed storage to use for trace buffers, consider the amount of central storage available.

COMASID
Traces activity in the common area page. This is the default.

NOCOMASID
Prevents tracing of activity in the common area page.

DMPREC
Includes trace data in the SVC dump requested when RSM enters recovery processing. The SYSRSM trace is suspended while the dump is in progress. The dump contains the most recent trace data recorded prior to the problem. With this dump option, which is a default:
- The trace tables are not dumped when RSM enters recovery processing.
- Tracing continues on other processors during recovery processing.

NODMPREC
Prevents trace data from being dumped if RSM enters recovery processing.

DMPOFF
Causes trace data to be dumped when tracing for RSM is turned off with a TRACE CT,OFF,COMP=SYSRSM command or with an OFF parameter in a CTnRSMxx parmlib member.

NODMPOFF
Prevents writing of a dump when the TRACE operator command is entered to stop the trace. This is the default.

Function Trace Options: Function trace options identify the RSM functions and services to be traced. The options are:

ASPCREAT
Traces events for the address space create function.

DFSTEAL
Traces events for the double frame steal function.

DIV
Traces all events in the data-in-virtual services group. The DIV option is equivalent to specifying DIVACCUN, DIVMAP, DIVMAPLV, DIVRES, DIVRESLV, DIVRTR, DIVSAVE, and DIVUNMAP. The group options, which can be specified separately, are:

DIVACCUN
Trace the DIV ACCESS and DIV UNACCESS services.

DIVMAP
Traces the data-in-virtual MAP service.

DIVMAPLV
Traces the data-in-virtual MAP service (with LOCVIEW=MAP on previous ACCESS).

DIVRES
Traces the data-in-virtual RESET service.

DIVRESLV
Traces the data-in-virtual RESET service (with LOCVIEW=MAP on previous ACCESS).

DIVRTR
Traces the data-in-virtual services router.

DIVSAVE
Traces the data-in-virtual SAVE service.

DIVUNMAP
Traces the data-in-virtual UNMAP service.

DSPCONV
Traces events in the data space convert interface function.

DSPLIMIT
Traces events in the data space limit interface function.

DATASPAC
Traces all events in the data space and hiperspace group. The DATASPAC option is equivalent to specifying DSPSERV and HSPSERV. The group options, which can be specified separately, are:

DSPSERV
Traces all events in the data space services group. This option is equivalent to specifying DSPCREAT, DSPDELET, DSPEXTEN, DSPIOOF, DSPIOON, DSPREL, and DSPSRTR. The group options, which can be specified separately, are:

DSPCREAT
Traces events in the DSPSERV CREATE service.

DSPDELET
Traces events in the DSPSERV DELETE service.

DSPDRFOF
Traces events in DSPSERV define DREF off.

DSPDRFON
Traces events in DSPSERV define DREF on.

DSPEXTEN
Traces events in the DSPSERV EXTEND service.

DSPLOAD
Traces events in the DSPSERV LOAD service.

DSPIOOF
Traces events in the DSPSERV IOOFF service.

DSPIOON
Traces events in the DSPSERV IOON service.

DSPOUT
Traces events in the DSPSERV OUT service.

DSPREL
Traces events in the DSPSERV RELEASE service.

DSPSRTR
Traces events in the DSPSERV router service.

DSPSRTRD
Traces events in the DSPSERV disabled RTR service.

HSPSERV
Traces all events in the hiperspace services group. This option is equivalent to specifying HSPCACHE and HSPSCROL. The group options, which can be specified separately, are:

HSPCACHE
Traces events in the HSPSERV cache services.

HSPSCROL
Traces events in the HSPSERV scroll services.

DUMPSERV
Traces the dumping function.

FAULTS
Traces all events in the fault services group. This option is equivalent to specifying FLTASP, FLTDSP, and FLTEPROT. The group options, which can be specified separately, are:

FLTASP
Traces all events in the address space faults group. This option is equivalent to specifying FLTADPAG, FLTAEPAG, and FLTAESEG. The group options, which can be specified separately, are:

FLTADPAG
Traces disabled address space page faults.

FLTAEPAG
Traces enabled address space page faults.

FLTAESEG
Traces enabled address space segment faults.

FLTAHPAG
Traces address space page faults for address above the 2 gigabytes bar.

FLTAHSEG
Traces address space segment faults for address above the 2 gigabytes bar.

FLTAREGN
Traces address space region faults.

FLTATYPE
Traces address space type faults.

FLTDSP
Traces all events on the data space faults group. This option is equivalent to specifying FLTDEN and FLTDDIS. The group options, which can be specified separately, are:

FLTDEN
Traces enabled data space faults.

FLTDDIS
Traces disabled data space faults.

FLTEPROT
Traces protection faults.

FREEFRAM
Traces the free frame function.

GEN
Traces all events in the general function group. This option is equivalent to specifying GENDEFER, GENIOCMP, and GENTERM. The group options, which can be specified separately, are:

GENDEFER
Traces general defers.

GENIOCMP
Trace general I/O completion.

GENTERM
Traces general abends.

IARVSERV
Traces all IARVSERV requests. The Virtual Services group options, which can be specified separately, are:

VSCHGACC
Traces IARVSERV CHANGEACCESS requests.

VSROUTR
Traces IARVSERV service router.

VSSHARE
Traces IARVSERV SHARE requests.

VSSHSEG
Traces IARVSERV SHARESEG requests.

VSUNSHAR
Traces IARVSERV UNSHARE requests.

IARV64
Traces all IARV64 requests. The High Virtual services group options, which can be specified separately are:

V6CHGURD
Traces IARV64 CHANGEGUARD requests

V6DETACH
Traces IARV64 DETACH requests

V6DISCAR
Traces IARV64 DISCARDDATA requests

V6GETSTR
Traces IARV64 GETSTOR requests

V6LIST
Traces IARV64 LIST requests

V6PAGFIX
Traces IARV64 PAGEFIX requests

V6PAGIN
Traces IARV64 PAGEIN requests

V6PAGOUT
TracesIARV64 PAGEOUT requests

V6PAGUNF
Traces IARV64 PAGEUNFIX requests

V6ROUTR
Traces IARV64 service router

MACHCHK
Traces the machine check function.

MIGRAT
Traces the migration function.

PER
Traces when RSM is entered as a result of a PER interrupt.

PGSER
Traces all events in the paging services group. This option is equivalent to specifying PGANY, PGFIX, PGFREE, PGLOAD, PGOUT, PGREL, and PRSRTR. The group options, which can be specified separately, are:

PGANY
Traces events in the page anywhere service.

PGFIX
Traces events in the page fix service.

PGFREE
Traces events in the page free service.

PGLOAD
Traces events in the page load service.

PGOUT
Traces events in the page out service.

PGREL
Traces events in the page release service.

PGSRTR
Traces events in the paging service routers.

QFSTEAL
Traces events for the quad frame steal function.

RECONFIG
Traces the reconfiguration function.

RPBPMGMT
Traces the RSM cell pool management function.

SUBSPACE
Traces all events in the subspace group. This option is equivalent to specifying SSPCONV and IARSUBSP. The group options, which can be specified separately, are:

SSPCONV
Traces the subspace conversion services.

IARSUBSP
Traces the subspace services group. This option is equivalent to specifying SSPIDENT, SSPCREAT, SSPASSIG, SSPUNAS, SSPDELET, SSPUNID, and SSPSRTR. The group options, which can be specified separately, are:

SSPIDENT
Traces the IARSUBSP IDENTIFY service.

SSPCREAT
Traces the IARSUBSP CREATE service.

SSPASSIG
Traces the IARSUBSP ASSIGN service.

SSPUNAS
Traces the IARSUBSP UNASSIGN service.

SSPDELET
Traces the IARSUBSP DELETE service.

SSPUNID
Traces the IARSUBSP UNIDENTIFY service.

SSPSRTR
Traces the IARSUBSP router.

STORMOD
Traces all events in the storage state modification group. This group is equivalent to specifying CLONEPAG and CLONESEG. The group options, which can be specified separately, are:

CLONEPAG
Traces the page table entry copied to a subspace.

CLONESEG
Traces the segment table entry copied to a subspace.

SWAP
Traces all events in the swap services group. This option is equivalent to specifying SWAPIN and SWAPOUT. The group options, which can be specified separately, are:

REALSWAP
Traces events during in-real swap processing.

SWAPIN
Traces events in the swap-in service.

SWAPOUT
Traces events in the swap-out service.

TRACE*
Traces the trace function. This function is always traced.

UIC
Traces the unreferenced interval count function.

VIO
Traces the virtual I/O function.

VR
Traces the V=R allocation function.

VSM
Traces all events in the VSM services group. This option is equivalent to specifying VSMFRMN and VSMGTMN. The group options, which can be specified separately are:

VSMFRMN
Traces events in the FREEMAIN service.

VSMGTMN
Traces events in the GETMAIN service.

XMPOST
Traces the cross memory posting function.

Event Trace Options: Event trace options identify the events for RSM to collect trace data. The options are:

ESTOR
Traces all events in the expanded storage management group. This option is equivalent to specifying ESDEQ, ESENQ, ESFREE, and ESGET. The group options, which can be specified separately, are:

ESGET
Traces get expanded storage.

ESENQ
Traces enqueue expanded storage.

ESDEQ
Traces dequeue expanded storage.

ESFREE
Traces free expanded storage.

FUNCREQ
Traces the function request event.

PAGEREQ
Traces all events in the page request group. This option is equivalent to specifying PAGEA2R, PAGEDEF, PAGEE2R, PAGEP2R, PAGEREL PAGER2A, PAGER2E, PAGER2EA, PAGER2P, PAGER2R and PAGETAPS. The group options, which can be specified separately, are:

PAGEA2R
Traces requests to move a page from auxiliary to central storage.

PAGEDEF
Traces requests to move a page to central storage was deferred for lack of a frame

PAGEE2R
Traces requests to move a page from expanded to central storage.

PAGEP2R
Traces requests to move a page from permanent to central storage.

PAGEREL
Traces requests related to I/O in-progress or related to a defer event.

PAGER2A
Traces requests to move a page from central storage to auxiliary storage.

PAGER2E
Traces requests to move a page from central storage to expanded storage.

PAGER2EA
Traces requests to start an asynchronous page move from central storage to expanded storage.

PAGER2P
Traces requests to move a page from central storage to permanent storage.

PAGER2R
Traces requests to move a page from central storage to central storage.

PAGETAPS
Traces requests to complete an asynchronous page move from central storage to expanded storage.

PGEVENTS
Traces all events in the page fix/free group. This option is equivalent to specifying FIX and FREE. The group options, which can be specified separately, are:

FIX
Traces a page being fixed.

FREE
Traces a page being freed.

REGIONGR
Traces all events in the region table group. This option is equivalent to specifying CREG1ST, CREG2ND, and CREG3RD. The group options, which can be specified separately, are:

CREG1ST
Traces the creation of a region 1st table.

CREG2ND
Traces the creation of a region 2nd table.

CREG3RD
Traces the creation of a region 3rd table.

RSTOR
Traces all events in the frame management group. This option is equivalent to specifying RSDEQ, RSENQ, RSFREE, RSGET and HVFRGRP. The group options, which can be specified separately, are:

HVFRGRP
Traces events for frame management of high virtual frames. This option is equivalent to specifying HVFRENQ, HVFRDEQ, HVPGTENQ, and HVPGTDEQ. The group options, which can be specified separately, are:

HVFRDEQ
Traces when a frame is dequeued from the high virtual frame queue.

HVFRENQ
Traces when a frame is enqueued onto the high virtual frame queue.

HVPGTDEQ
Traces when a frame is dequeued from the high virtual page table frame queue.

HVPGTENQ
Traces when a frame is enqueued onto the high virtual page table frame queue.

RSDEQ
Traces all events in the dequeue frame group. This option is equivalent to specifying RSDDEFER, RSDFIX, RSDPAG, RSDSBUF, RSDSQA, and RSDVRW. The group options, which can be specified separately, are:

RSDDEFER
Traces when a frame is dequeued from the deferred FREEMAIN frame queue or the orphan frame queue.

RSDFIX
Traces when a frame is dequeued from the fixed frame queue or the local quad frame queue.

RSDGDFER
Traces when a frame is dequeued from the general defer frame queue.

RSDPAG
Traces when a frame is dequeued from the pageable frame queue.

RSDSBUF
Traces when a frame is dequeued from the central storage buffer frame queue.

RSDSQA
Traces when a frame is dequeued from the SQA frame queue.

RSDVRW
Traces when a frame is dequeued from the V=R waiting frame queue.

RSENQ
Traces all events in the enqueue frame group. This option is equivalent to specifying RSEDEFER, RSEFIX, RSEPAG, RSESBUF, RSESQA, and RSEVRW. The group options, which can be specified separately, are:

RSEDEFER
Traces when a frame is enqueued onto the deferred or orphan frame queue.

RSEFIX
Traces when a frame is enqueued onto to the fixed frame queue or the local quad frame queue.

RSEPAG
Traces when a frame is enqueued onto to the pageable frame queue.

RSESBUF
Traces when a frame is enqueued onto to the central storage buffer frame queue.

RSEGDFER
Traces when a frame is enqueued on the general defer frame queue.

RSESQA
Traces when a frame is enqueued onto to the SQA frame queue.

RSEVRW
Traces when a frame is enqueued onto to the V=R waiting frame queue.

RSFREE
Traces all events in the free frame group. This option is equivalent to specifying RSFDBL and RSFSNG. The group options, which can be specified separately, are:

QFFREE
Traces when a quad group is freed.

QHFREE
Traces when a quad holding frame is freed.

QSFREE
Traces when a single quad frame is freed.

RSFDBL
Traces when a double frame is freed.

RSFSNG
Traces when a single frame is freed.

RSGET
Traces all events in the get frame group. This option is equivalent to specifying RSGDBL and RSGSNG. The group options, which can be specified separately, are:

QFGET
Traces when a quad group is gotten.

QHGET
Traces when a quad holding frame is gotten.

QSGET
Traces when a single quad frame is gotten.

RSGDBL
Traces when a double frame is gotten.

RSGSNG
Traces when a single frame is gotten.

SHRDATA
Traces all events in the IARVSERV services group. This option is equivalent to specifying GRPCREAT, GRPDEL, GRPPART, VIEWADD, VIEWCHG, VIEWDEL, and VIEWMOVE. The group options, which can be specified separately, are:

GRPCREAT
Traces the creation of new sharing groups.

GRPDEL
Traces the deletion of existing sharing groups.

GRPPART
Traces the partitioning of existing sharing groups.

VIEWADD
Traces the addition of views to sharing groups.

VIEWCHG
Traces the changing of storage attributes of the view.

VIEWDEL
Traces the deletion of views from sharing groups.

VIEWMOVE
Traces the move of existing views from one sharing group to another.

TRACEB
Traces the trace buffer event. This event is always traced.

WORKUNIT
Traces all events in the net event trace group. This option is equivalent to specifying SUSPEND and RESUME. The group options, which can be specified separately, are:

ENABLE
Traces requests to enable a unit of work.

SUSPEND
Traces requests to suspend a unit of work.

RESUME
Traces requests to resume a unit of work.

XEPLINK
Traces all events in the external entry point linkage group. This option is equivalent to specifying XEPENTRY and XEPEXIT. The group options, which can be specified separately, are:

XEPENTRY
Traces entry to the entry point.

XEPEXIT
Traces exit from the entry point.

Examples of Requesting SYSRSM Traces

Example 1: CTnRSMxx Member

The member requests tracing of the FAULTS services group, the PGANY service, and the VIO function, but only for address spaces X'11' and X'41' and for job PGM1.

```
TRACEOPTS
   ON
   ASID(11,41)
   JOBNAME(PGM1)
   OPTIONS('FAULTS','PGANY','VIO')
```

Example 2: TRACE Command

The example specifies that options are to be obtained from the parmlib member CTWRSM17.

```
trace ct,on,comp=sysrsm,parm=ctwrsm17
```

Example 3: TRACE Command

The example requests the same trace as Example 2, but specifies all options in the REPLY.

```
trace ct,on,comp=sysrsm
* 78 ITT006A ...
reply 78,options=(faults,pgany,vio),asid=(11,41),jobname=(pgm1),end
```

Formatting a SYSRSM Trace

Format the trace with an IPCS CTRACE COMP(SYSRSM) subcommand. The subcommand has no OPTIONS values.

Output from a SYSRSM Trace

CTRACE COMP(SYSRSM) FULL Subcommand Output

The following is an example of RSM component trace records formatted with the CTRACE COMP(SYSRSM) FULL subcommand:

```
 XEPENTRY  00000001  18:18:06.441411  External Entry Point Entry
      FUNC1... PGFIX             Page Fix
      JOBN1... ASNB     ASID1... 000D     PLOCKS.. 80000001 CPU..... 0000
      JOBN2... ASNB     ASID2... 000D     RLOCKS.. 80000000
      KEY..... 0036     ADDR.... 015FF008 ALET.... 00000000
      7D00
      KEY..... 002C     ADDR.... 005FEF80 ALET.... 00000000
      00000000 005EF000 005EFFFF 00F00200 00FF8DD8 00F8BD00 00F8BC98
      005EFFC3 005EF02C 00F8BD50 005EF02C 00F8BE00 00F8BC00 005FEF78
      00FF91DA 81082798


FIX       00000003  18:18:06.441921  Page Being Fixed
      FUNC1... PGFIX             Page Fix
      JOBN1... ASNB     ASID1... 000D     PLOCKS.. 88004001 CPU..... 0000
      JOBN2... ASNB     ASID2... 000D     RLOCKS.. 88004000
      KEY..... 0036     ADDR.... 015FF008 ALET.... 00000000
      7D003500
      KEY..... 005D     ADDR.... 005EF000 ALET.... 00000000
      KEY..... 0001     ADDR.... 011F8220 ALET.... 00000000
      0151182C 012D2E60 81C00000 03000001 0000000D 005EF000 00000000 00000000


XEPENTRY  00000001  18:18:06.461716  External Entry Point Entry
      FUNC1... PGFREE            Page Free
      JOBN1... ASNB     ASID1... 000D     PLOCKS.. 80000001 CPU..... 0000
      JOBN2... ASNB     ASID2... 000D     RLOCKS.. 80000000
      KEY..... 0036     ADDR.... 015FF008 ALET.... 00000000
      8100
      KEY..... 002C     ADDR.... 005FEF80 ALET.... 00000000
      005F5EC0 00F8BC08 00000000 00F00200 00FF8DD8 00FF6896 00F8DB00
      00000008 00F8BF2C 00FF7B60 00000C60 00F8BE00 00F8Bc00 005FEF78
      00FF96A4 810801B8
```

```
FREE      00000004  18:18:06.461766  Page Being Freed
      FUNC1... PGFREE            Page Free
      JOBN1... ASNB     ASID1... 000D     PLOCKS.. 88004001 CPU..... 0000
      JOBN2... ASNB     ASID2... 000D     RLOCKS.. 88004000
      KEY..... 0036     ADDR.... 015FF008 ALET.... 00000000
      8100
      KEY..... 005D     ADDR.... 005F5000 ALET.... 00000000
      KEY..... 0001     ADDR.... 01210660 ALET.... 00000000
      011F9CA0 01242560 81C00000 03000000 0000000D 005F5000 00000000 00000000


FREE      00000004  18:18:06.461805  Page Being Freed
      FUNC1... PGFREE            Page Free
      JOBN1... ASNB     ASID1... 000D     PLOCKS.. 88004001 CPU..... 0000
      JOBN2... ASNB     ASID2... 000D     RLOCKS.. 88004000
      KEY..... 0036     ADDR.... 015FF008 ALET.... 00000000
      8100
      KEY..... 005D     ADDR.... 005EF000 ALET.... 00000000
      KEY..... 0001     ADDR.... 011F8220 ALET.... 00000000
      0151182C 012D2E60 81C00000 03000000 0000000D 005EF000 00000000 00000000
```

The fields that you may need in the report are:

FUNC1
The function in control at the time the trace event was recorded.

JOBN1
The job name identifying the address space that contains the unit of work requesting the RSM service.

JOBN2
The job name that matched a name in the job name list provided with the TRACE operator command.

ASID1
The ASID identifying the address space that contains the unit of work requesting the RSM service.

ASID2
The ASID that matched an identifier in the ASID list provided with the TRACE operator command.

CPU
The central processor identifier for the processor the trace is running on.

SYSSPI Component Trace

Before Using This Component Trace

This topic assumes you have read:
- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSSPI component trace for the service processor interface (SPI).

Information	For SYSSPI:
Parmlib member	None

Information	For SYSSPI:
Default tracing	No
Trace request OPTIONS parameter	None
Buffer	• Default: 64KB • Range: N/A • Size set by: MVS system • Change size after IPL: No • Location: In the component area
Trace records location	Address-space buffer
Request of SVC dump	By the component
Trace formatting by IPCS	CTRACE COMP(SYSSPI)
Trace format OPTIONS parameter	None

Requesting a SYSSPI Trace

Request a SYSSPI trace at the direction of the IBM Support Center. Do the following:

1. Start the trace with the command:
   ```
   TRACE CT,ON,COMP=SYSSPI
   ```
2. After the interval specified by IBM, stop the trace with the command:
   ```
   TRACE CT,OFF,COMP=SYSSPI
   ```

When the buffer fills up, the component requests an SVC dump, which includes the contents of the buffer. Optionally, the operator could enter a DUMP command.

Formatting a SYSSPI Trace

1. Use superzap to change a module. IBM supplies the change.
2. Format the trace with an IPCS CTRACE COMP(SYSSPI) subcommand. The subcommand has no options values.

SYSTTRC Transaction Trace

Transaction trace does not participate in component trace-controlled processing. It is a standalone tracing facility. Do not use trace CT commands for transaction trace. Do not attempt to add a component trace parmlib member for transaction trace. For information on transaction trace, see Chapter 12, “Transaction Trace” on page 12-1.

SYSVLF Component Trace

Before Using This Component Trace

This topic assumes you have read:
- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSVLF component trace for the virtual lookaside facility (VLF).

Information	For SYSVLF:
Parmlib member	None
Default tracing	Yes; minimal; unexpected events
Trace request OPTIONS parameter	None
Buffer	• Default: N/A • Range: N/A • Size set by: MVS system • Change size after IPL: No • Location: Data space. Enter DISPLAY J,VLF to identify the VLF data spaces. In the REPLY for the DUMP command, specify DSPNAME=('VLF'.Dclsname,'VLF'.Cclsname), where *clsname* is a VLF class name.
Trace records location	Address-space buffer, data-space buffer
Request of SVC dump	By DUMP or SLIP command or when SYSVLF full tracing is turned off
Trace formatting by IPCS	CTRACE COMP(SYSVLF)
Trace format OPTIONS parameter	None

Requesting a SYSVLF Trace

A minimal trace runs whenever VLF is in control. No actions are needed to request the minimal trace.

To record more than the minimal trace, request full tracing with the TRACE operator command. Note that full tracing can slow system performance. The following table indicates the parameters you can specify on a TRACE CT command. In response to the command, the system does not prompt the operator for a reply.

Parameters	Allowed on TRACE CT for Trace?
ON or OFF	One is required
nnnnK or nnnnM	No
COMP	Required
SUB	No
PARM	No

When you turn the full tracing off, the system writes a dump containing the trace records, then resumes minimal tracing.

Examples of Requesting and Stopping SYSVLF Full Traces

Example 1: Requesting a SYSVLF Full Trace

The command requests a full trace.

```
TRACE CT,ON,COMP=SYSVLF
```

Example 2: Stopping a SYSVLF Full Trace

The command turns off full tracing. In response, the system writes a dump and resumes minimal SYSVLF tracing.

```
TRACE CT,OFF,COMP=SYSVLF
```

Example 3: Command for SYSVLF Tracing in a Sysplex

The following command turns on tracing for a SYSVLF trace in the systems of a sysplex. Because SYSVLF has no parmlib member, the CTIITT00 member is used to prevent prompts.

```
route *all,trace ct,on,comp=sysvlf,parm=ctiitt00
```

Formatting a SYSVLF Trace

Format the trace with an IPCS CTRACE COMP(SYSVLF) subcommand. The subcommand has no OPTIONS values.

Output from a SYSVLF Trace

CTRACE COMP(SYSVLF) FULL Subcommand Output

The following is an example of VLF component trace records formatted with the CTRACE COMP(SYSVLF) FULL subcommand. It shows formatted exception records from the trace buffers.

```
                   VLF  COMPONENT TRACE FULL FORMAT
 **** 01/27/90
 COFRCVRY  00000000  16:03:02.181262  VLF RECOVERY ENTRY
   HASID... 000E     SASID... 000E     CPUID... FF170284 30900000
   MODNAME. COFMPURG ABEND... 840C4000 REASON.. 00000011
   EPTABLE. PURG  ESTA  ....  ....  ....  ....  ....  ....  ....  ....
 COFRCVRY  00000001  16:03:02.181324  VLF RECOVERY EXIT
   HASID... 000E     SASID... 000E     CPUID... FF170284 30900000
   MODNAME. COFMPURG ABEND... 840C4000 REASON.. 00000011
   RETCODE. 00000000 RSNCODE. 00000000 FTPRTS.. 80300000 DATA.... 00000000
 .
 .
 .
```

The following explains fields in the report. Additional fields that are not shown in the example can be in a report. These additional fields are explained 11-124.

COFRCVRY
: The name or identifier of the trace record.

00000000
: The identifier in hexadecimal

16:03:02.181262
: The time stamp indicating when the record was placed in the trace table

HASID... 000E
: The home address space identifier

SASID... 000E
: The secondary address space identifier

CPUID... FF170284 30900000
The identifier of the processor that placed the record in the trace table

CALLER
The address of the routine that issued a VLF service request, such as DEFINE, CREATE, NOTIFY, PURGE, etc..

MODNAME. COFMPURG
The name of the module that was running

ABEND... 840C4000
The abend that occurred and caused VLF to enter recovery is 0C4

REASON.. 00000011
The reason code associated with the abend

EPTABLE. PURG ESTA
Information used for diagnosis by IBM

RETCODE. 00000000
The return code that was issued by the module that is exiting

RSNCODE. 00000000
The reason code that was issued by the module that is exiting

FTPRTS.. 80300000
Information used for diagnosis by IBM

DATA.... 00000000
Information used for diagnosis by IBM

Other Fields: Fields that are not shown in the example CTRACE output but that may appear in a report are:

CINDX
The concatenation index of the major name for which an object has been created or retrieved

CLASS... NPDS3
The name of a VLF class

DDNAME
The DDNAME of the concatenated data set list

FUNC=xxxx
Indication of the function for which a NOTIFY occurred

FUNCCODE
The hexadecimal value of the NOTIFY function code when it cannot be interpreted

MAJOR
The major name

MINADDR
Address of a field containing a minor name

MINALET
Access list entry token (ALET) associated with the address used to locate the minor name

MINOR
The minor name

OBJSIZE
The total size, in bytes, of the object returned by a COFRETRI macro

PARMS
Hexadecimal dump of the COFNOTIF macro parameter list

TLSTADDR
Address of a target list for a COFRETRI macro

TLSTALET
Access list entry token (ALET) of a target list for a COFRETRI macro

TLSTSIZE
The length, in bytes, of the target list

UTOKEN
User token returned by a COFIDENT macro and required as input for COFREMOV, COFCREAT, and COFRETRI macros

VOLSER
The volume serial

SYSWLM Component Trace

Before Using This Component Trace

This topic assumes you have read:

- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSWLM component trace for the workload manager (WLM).

Information	For SYSWLM:
Parmlib member	None
Default tracing	Yes; minimal; unexpected events
Trace request OPTIONS parameter	None
Buffer	• Default: 64KB • Range: 64KB - 16M • Size set by: MVS system • Change size after IPL: Yes, when starting a trace • Location: Extended common service area (ECSA)
Trace records location	Address-space buffer, trace data set
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSWLM)
Trace format OPTIONS parameter	None

Requesting a SYSWLM Trace

Request a SYSWLM component trace by a TRACE CT command.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON, nnnnK, OFF	One is required. nnnnK specifies the size of the buffer.
COMP	Required
SUB	No
PARM	No

Parameters	Allowed on TRACE CT for Writer?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	No
JOBNAME	No
OPTIONS	No
WTR	Yes

Examples of Requesting SYSWLM Traces

The following is an example of requesting a SYSWLM component trace.

```
trace ct,on,comp=syswlm
* 17 ITT006A ...
reply 17,end
```

Formatting a SYSWLM Trace

Format the trace with an IPCS CTRACE COMP(SYSWLM) subcommand. The subcommand has no OPTIONS values.

Output from a SYSWLM Trace

CTRACE COMP(SYSWLM) SHORT Subcommand Output

The following is an example of SYSWLM component trace records formatted with the CTRACE COMP(SYSWLM) SHORT subcommand.

```
MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
--------  --------  ---------------  -----------

WLMEPEXT  00005004  14:07:42.807618  Entry Point Exited
WLMEPENT  00005003  14:07:46.335563  Entry Point Entered
WLMEPEXT  00005004  14:07:46.336376  Entry Point Exited
WLMEPENX  00005005  14:07:46.336540  Entry Point Entered Exception
WLMEPENT  00005003  14:07:46.336557  Entry Point Entered
WLMEPENT  00005003  14:07:46.337909  Entry Point Entered
SMSYNMEM  00000921  14:07:46.512018  SM Synch XCF Member
WLMEPEXT  00005004  14:07:46.512360  Entry Point Exited
WLMEPENT  00005003  14:07:46.512374  Entry Point Entered
SMSYNMEM  00000921  14:07:46.594486  SM Synch XCF Member
```

CTRACE COMP(SYSWLM) FULL Subcommand Output

The following is an example of SYSWLM component trace records formatted with the CTRACE COMP(SYSWLM) FULL subcommand.

```
SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
-------   --------  --------  ---------------  -------------
SY1       WLMEPENT  00005003  14:52:23.449339  Entry Point Entered
     FUNCID... 0409                CPU...... 0001
     HOMEASID. 000B                HJOBNAME. WLM
     REQASID.. 0000                RJOBNAME. UNKNOWN
     KEY...... 5018 RUCA_EPIDS IWMDMPRP
         04098000
     KEY...... 501E PARM1
         00000084
     KEY...... 501F PARM2
         00000040
     KEY...... 5020 PARM3
         00000000
```

The following explains fields in the report.

FUNCID
The module table entry for the module that wrote the trace record.

CPU
The CPU that the module was running on.

HOMEASID
ASID from PSAAOLD.

REQASID
ASID that was explicitly coded on trace invocation.

HJOBNAME
JOBNAME of home address space.

RJOBNAME
JOBNAME that was explicitly coded on trace invocation.

KEY
Identifies the type of data that follows. The data is formatted in both HEX and EBCDIC.

SYSXCF Component Trace

Before Using This Component Trace

This topic assumes you have read:
- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSXCF component trace for the cross-system coupling facility (XCF).

Information	For SYSXCF:
Parmlib member	CTnXCFxx Default member: CTIXCF00 specified in COUPLE00 member
Default tracing	Yes; minimal; unexpected events
Trace request OPTIONS parameter	In CTnXCFxx or REPLY for TRACE command

Information	For SYSXCF:
Buffer	• Default: 72KB • Range: 16KB - 16MB (System rounds size up to a multiple of 72 bytes.) • Size set by: CTnXCFxx member • Change size after IPL: No • Location: Extended local system queue area (ELSQA) of XCFAS
Trace records location	Address-space buffer, trace data set
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSXCF)
Trace format OPTIONS parameter	Yes

Requesting a SYSXCF Trace

Specify options for requesting a SYSXCF component trace on a CTnXCFxx parmlib member or on the reply for a TRACE CT command.

If you specify additional tracing options while the system is running, place the trace records in a trace data set or sets, because the trace buffer size specified at initialization cannot be changed while the system is running. Specify NOWRAP to keep from losing trace records.

Note: NOWRAP prevents trace records written to the data set or sets from being overwritten. Once the data set or sets are filled, no more records are written to them. The system still writes trace records to the address-space buffers. The system wraps the address-space buffers, so that trace records may be lost. Be sure to allocate enough space on the data set or sets to hold all the records needed for diagnosis.

CTnXCFxx Parmlib Member

The following table indicates the parameters you can specify on a CTnXCFxx parmlib member.

Parameters	Allowed on CTnXCFxx?
ON or OFF	Yes
ASID	No
JOBNAME	No
BUFSIZE	Yes, at IPL or when reinitializing XCF
OPTIONS	Yes
SUB	No
PRESET	No
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

Note: You can change the buffer size only at IPL or when reinitializing XCF. Specify the new buffer size in the BUFSIZE parameter on the CTnXCFxx member being used.

IBM supplies the CTIXCF00 parmlib member, which specifies the XCF tracing begun at initialization. The contents of CTIXCF00 are:

```
TRACEOPTS
   ON
   BUFSIZE(72K)
```

These parameters turn on the minimal XCF tracing and establish a trace buffer of 72KB. This member activates the minimal trace at initialization. In the IBM-supplied COUPLE00 parmlib member, the CTRACE parameter specifies CTIXCF00 as the default.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON or OFF	One is required
nnnnK or nnnnM	No
COMP	Required
SUB	No
PARM	Yes

Parameters	Allowed on TRACE CT for Write?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	No
JOBNAME	No
OPTIONS	Yes
WTR	Yes

OPTIONS Parameter

The values for the OPTIONS parameter for the CTnXCFxx parmlib member and reply for a TRACE command, in alphabetical order, are:

ARM
: Traces events for automatic restart management services.

CFRM
: Traces events for coupling facility resource management services.

GROUP
: Traces events for group services, such as XCF groups joining or disassociating from XCF services.

GRPNAME=(*groupname***[,***groupname***]...)**
: Reduces tracing to events for only the specified XCF groups. If GRPNAME is specified, the GROUP, SERIAL, SIGNAL, and STATUS options are filtered by the specified XCF group or groups; the STORAGE option is not filtered by GRPNAME. You can specify up to 8 XCF groups.

SERIAL
: Traces events for serialization services.

SFM
Traces events for sysplex failure management services.

SIGNAL
Traces events for signalling services processing.

STATUS
Traces events for XCF monitoring services and sysplex partitioning services.

STORAGE
Traces events for storage management services.

Examples of Requesting SYSXCF Traces

Example 1: CTnXCFxx Member

The member requests STORAGE and SIGNAL options. To minimize lost trace data, the member also starts external writer WTRDASD1 with the NOWRAP option specified and connects the trace to the writer.

```
TRACEOPTS
   WTRSTART(WTDASD1) NOWRAP
   ON
   WTR(WTDASD1)
   OPTIONS('STORAGE','SIGNAL')
```

Example 2: TRACE Commands

This example requests the same trace as Example 1.

```
trace ct,wrtstart=wtdasd1
trace ct,on,comp=sysxcf
* 62 ITT006A ...
r 62,wtr=wtdasd1,options=(storage,signal),end
```

Formatting a SYSXCF Trace

Format the trace with an IPCS CTRACE COMP(SYSXCF) subcommand. The OPTIONS parameter specifies the options that select trace records to be formatted. Your formatting options depend to a great extent on the tracing options you requested. Use the options to narrow down the records displayed so that you can more easily locate any errors. If the CTRACE subcommand specifies no options, IPCS displays all the trace records.

The options are:

ARM
Formats trace records for automatic restart management services.

CFRM
Formats trace records for coupling facility resource management services.

GROUP
Formats trace records for XCF group services, such as groups joining or disassociating from XCF services.

SERIAL
Formats trace records for serialization services.

SFM
Formats trace records for sysplex failure management services.

SIGNAL
Formats trace records for signalling services processing.

STATUS
Formats trace records for XCF monitoring services and sysplex partitioning services.

STORAGE
Formats trace records for storage management services.

Output from a SYSXCF Trace

CTRACE COMP(SYSXCF) FULL Subcommand Output

The following is an example of SYSXCF component trace records formatted with the CTRACE COMP(SYSXCF) FULL subcommand.

```
                   SYSXCF  COMPONENT TRACE FULL FORMAT
**** 01/15/90
STORAGE   0F030001  21:01:30.893078  CROSS SYSTEM COUPLING FACILITY
   00000000  00000004  00000000  | ................ |
   00000000  00000000  | ................ |
   00000000  00000000  | ................ |
   00000000  0000                        | ......           |
SIGNAL    08560000  21:01:30.941115  CROSS SYSTEM COUPLING FACILITY
   01000000  08018000  826445C0  C9D5C9E3  | ........b..{INIT |
   40404040  00000000  00000000  |     ............ |
   00000000  7F68A810  00000000  | ....".y......... |
   00000000  0000                        | ......           |
.
.
.
```

SYSXES Component Trace

Before Using This Component Trace

This topic assumes you have read:

- “Planning for Component Tracing” on page 11-3
- “Obtaining a Component Trace” on page 11-10
- “Viewing the Component Trace Data” on page 11-23

The following summarizes information for requesting a SYSXES component trace for cross-system extended services (XES).

Information	For SYSXES:
Parmlib member	CTnXESxx Default member: CTIXES00 specified in COUPLE00 member
Default tracing	Yes; minimal; unexpected events
Trace request OPTIONS parameter	In CTnXESxx or REPLY for TRACE command

Information	For SYSXES:
Buffer	• Default: 168KB • Range: 16KB - 16MB • Size set by: CTnXESxx member or TRACE CT command • Change size after IPL: Yes • Location: Data space. In the REPLY for the DUMP command, specify SDATA=XESDATA and DSPNAME=(*asid*.IXLCTCAD) where *asid* is the ASID for address space XCFAS
Trace records location	Data-space buffer, trace data set
Request of SVC dump	By DUMP or SLIP command
Trace formatting by IPCS	CTRACE COMP(SYSXES)
Trace format OPTIONS parameter	Yes

SYSXES supports sublevel tracing. Tracing options are inherited through a hierarchy of trace levels. If you set trace options without specifying a sublevel, the options apply at the highest level, or head, of the hierarchy. A sublevel inherits its trace options from the next higher level, unless options are specified explicitly for the sublevel. If you set trace options for a sublevel, the options are inherited by any sublevels lower in the hierarchy.

Figure 11-9 is shows the hierarchical structure of SYSXES traces.

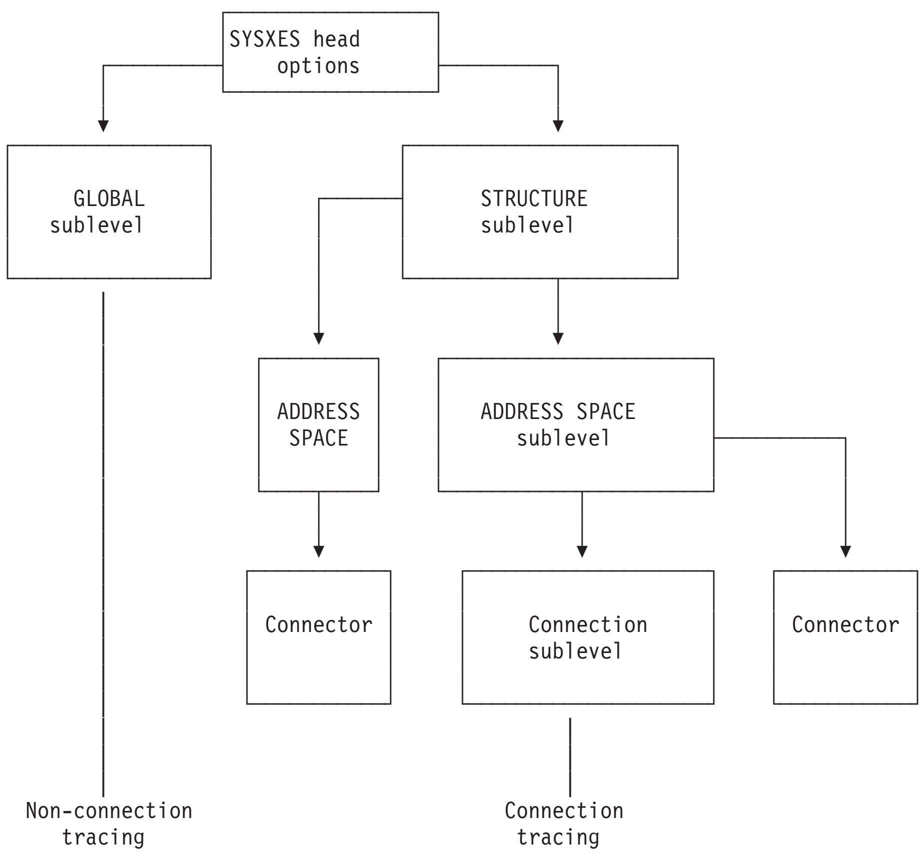

Figure 11-9. SYSXES SUB Trace Structure

Two classes of sublevel traces inherit the head trace options:

- The global sublevel trace has its own trace buffer and controls tracing that is not related to any particular connection. Request GLOBAL tracing by specifying SUB=(GLOBAL).
- Connection sublevels control tracing for a particular connection. Each connection has its own trace buffer. Connection sublevels are filtered hierarchically based on:
  1. Structure name (STRNAME) of the coupling facility structure to which the system is connected
  2. Address space identifier (ASID) of the address space from which the connection was made
  3. Connection name (CONNAME) of the particular connector for which tracing is requested

  Therefore, options specified for a particular structure name are inherited only by address spaces connected to that structure. Options specified for a particular address space are inherited only by connections that are connected through that address space.

  Specify **SUB=(strname)**, **SUB=(strname.asid)**, or **SUB=(strname.asid.conname)**, depending on the degree of specificity you need. Do not specify **conname** without specifying **asid**. Also, do not specify **asid** without specifying **strname**. When specifying a structure name or connector name on the SUB option, if the name contains special characters, it must be in quotes. If it is in quotes, upper and lower case characters are not the same. Therefore the case information is important and must identically match the name used by the system. Once the name is enclosed in quotes it becomes case sensitive.

Requesting a SYSXES Trace

Specify options for requesting a SYSXES component trace on a CTnXESxx parmlib member or on the reply for a TRACE CT command.

CTnXESxx Parmlib Member

The following table indicates the parameters you can specify on a CTnXESxx parmlib member.

Parameters	Allowed on CTnXESxx?
ON or OFF	Yes
ASID	No
JOBNAME	No
BUFSIZE	Yes
OPTIONS	Yes
SUB	Yes, but only for a sublevel trace
PRESET	Yes, but only for a sublevel trace
LIKEHEAD	No
WTR	Yes
WTRSTART or WTRSTOP	Yes

Note: Buffer size must be specified in the parmlib member used at IPL. The buffer size can be modified by a TRACE command or by a parmlib member activated while the system is running.

Setting Buffer Size: To select a size for your trace buffers, consider the following:

- The trace buffers can be smaller if you are using an external writer, because buffer wrapping is not a concern.
- When re-creating a problem, you might first want to make the buffer size larger.
- SYSXES has one trace buffer of the specified size **per connector**, plus one for the global trace. The amount of storage used can be significant if the system is going to have many connectors. Furthermore, the trace buffers are allocated from a single common area data space (CADS). If the entire CADS is used up, subsequent connections will not be traced because buffer space is not available.
- The SYSXES trace buffers are in disabled reference (DREF) data space storage, so storage constraints may limit buffer size.

Changing Buffer Size: To change the size of your trace buffers while a trace is running, either issue a TRACE CT command or activate a different CTnXESxx parmlib member. You can use these methods to change SUB levels in the hierarchy so that different SUB traces can have different sized buffers.

TRACE and REPLY Commands

The following tables indicate the parameters you can specify on TRACE CT commands and a REPLY.

Parameters	Allowed on TRACE CT for Trace?
ON or OFF	One is required
nnnnK or nnnnM	Yes
COMP	Required
SUB	Yes
PARM	Yes

Parameters	Allowed on TRACE CT for Writer?
WTRSTART or WTRSTOP	One is required, if a writer is being used

Parameters	Allowed on REPLY for Trace?
ASID	No
JOBNAME	No
OPTIONS	Yes
WTR	Yes

OPTIONS Parameter

The values for the OPTIONS parameter for the CTnXESxx parmlib member and reply for a TRACE command, in alphabetical order, are:

ALL
Traces events listed for all of the options.

CONFIG
Traces changes in the state of connectivity to the coupling facility, such as addition or removal of paths.

CONNECT
Traces events for system and subsystem components that connect to or disconnect from XES resources and for exit processing.

HWLAYER
Traces events for the XES services that handle communications with the coupling facility.

LOCKMGR
Traces events related to global management of resources and to global management-related exits.

RECOVERY
Traces events within the modules that handle XES resource access failures, for both resource allocation and mainline command processing. This option provides more details than is provided by default.

REQUEST
Traces events related to requests to access data through XES mainline services.

SIGNAL
Traces events related to XES internal signalling.

STORAGE
Traces events related to management of XES control blocks.

Examples of Requesting SYSXES Traces

Example 1: CTnXESxx Member

The member requests a trace of HWLAYER, LOCKMGR, CONNECT, and REQUEST trace events and a buffer size of 100KB.

```
TRACEOPTS
   ON
   OPTIONS('HWLAYER','LOCKMGR','CONNECT','REQUEST')
   BUFSIZE(100K)
```

Example 2: TRACE Command

The example requests a trace of CONNECT, CONFIG, and STORAGE trace events for connection CON3 in ASID 5 for structure STR3.

```
trace ct,on,comp=sysxes,sub=(str3.asid(5).con3)
* 17 ITT006A ...
reply 17,options=(connect,config,storage),end
```

Formatting a SYSXES Trace

Format the trace with an IPCS CTRACE COMP(SYSXES) subcommand. The OPTIONS parameter specifies the options that select trace records to be formatted. Your formatting options depend to a great extent on the tracing options you requested. Use the options to narrow down the records displayed so that you can more easily locate any errors. If the CTRACE subcommand specifies no options, IPCS displays all the trace records.

ALL
Formats all trace records.

CONFIG
Formats changes in the state of connectivity to the coupling facility.

CONNECT
Formats events for system and subsystem components that connect to or disconnect from XES resources and for exit processing.

HWLAYER
Formats events for the XES services that handle communications with the coupling facility.

LOCKMGR
Formats events related to global management of resources and to global management-related exits.

RECOVERY
Formats events within the modules that handle XES resource access failures.

REQUEST
Formats events related to requests to access data through XES mainline services.

SIGNAL
Formats events related to XES internal signalling.

STORAGE
Formats events related to management of XES control blocks.

In the CTRACE subcommand, the SUB((subname.subname.subname)) parameter specifies the sublevel traces. A subname is:

- **GLOBAL** for an event not related to a particular connection.
- **strname.asid.conname** for an event related to the specified connector. The subname for a connection-related sublevel can contain up to three parts:
  - **strname** for the structure
  - **asid** for the address space identifier (ASID)
  - **conname** for the connection name, if **asid** is also specified

Any sublevel specification is valid for the QUERY option; for example:

```
SUB(STR3)
SUB(STR3.ASID(5))
```

Only the GLOBAL and fully qualified connection sublevel specifiers are valid with a COMP parameter; for example:

```
SUB(GLOBAL)
SUB(STR3.ASID(5).CON3)
```

Output from a SYSXES Trace

CTRACE COMP(SYSXES) SHORT Subcommand Output

The following is an example of SYSXES component trace records formatted with the following subcommand:

```
CTRACE COMP(SYSXES) SUB((GLOBAL)) SHORT OPTIONS((CONNECT,HWLAYER))
```

```
COMPONENT TRACE SHORT FORMAT
COMP(SYSXES)  SUBNAME((GLOBAL))
**** 10/20/93

MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
--------  --------  ---------------  -----------

HWLAYER   090C0002  20:47:22.096016  EXIT FROM IXLMLTAM
HWLAYER   07140001  20:47:22.096296  ENTRY TO IXLERTRR
HWLAYER   07140002  20:47:22.096429  EXIT FROM IXLERTRR
CONNECT   08190001  20:47:30.171676  CONNECTOR DIE ROUTINE
HWLAYER   090C0001  20:47:30.171718  ENTRY TO IXLMLTAM
HWLAYER   09030001  20:47:30.171758  ENTRY TO IXLMLXRB
HWLAYER   09080001  20:47:30.171779  ENTRY TO IXLM2SR START IMMED RE
HWLAYER   09080003  20:47:30.171804  ISSUING A SMSG COMMAND
CONNECT   08110001  20:47:30.172316  MAINLINE TIMER EXIT ENTERED
CONNECT   08110004  20:47:30.172476  MAINLINE TIMER EXITED
HWLAYER   09080004  20:47:30.180754  COMPLETION OF A SMSG COMMAND
```

Chapter 12. Transaction Trace

Transaction trace is like dropping a trail of bread crumbs. No matter where your transaction travels through the sysplex, you can always find it.

Transaction trace provides a consolidated trace of key events for the execution path of application- or transaction-type *work units* running in a multi-system application environment. By tracing the path of a work unit running in a single system, or (more importantly) across systems in a sysplex environment, that is being processed by multi-system transaction servers, subsystem interfaces, and resource managers, transaction trace enables a system programmer to debug problems in those environments.

The essential task of transaction trace is to aggregate data showing the flow of work between components in the sysplex that combine to service a transaction. Transaction trace traces events such as component entry, exit, exceptions and major events such as COMMIT, and ROLLBACK. Transaction trace should not be used as a component tracing facility.

The following topics explain transaction trace in detail:

- “How Transaction Trace Works”
- “Transaction Trace Commands” on page 12-2
- “Using IPCS To View Transaction Trace Output” on page 12-4

How Transaction Trace Works

Transaction trace (TTrace) is attached as a daughter task in the system trace address space, after master scheduler initialization completes. Once initialization has completed, and the first transaction trace command is entered with a filter that specifies the attributes of the work unit(s) to be traced, transaction trace is activated. Additional information, such as the use of an external writer, can also be specified for transaction trace processing.

Once transaction trace is activated, WLM Classify invokes a filter exit to determine whether the current work unit should be traced. The work unit’s attributes are compared with the command filter attributes to determine if tracing should occur. If tracing is required, a non-zero token is built and returned to the Classify caller. The transaction trace token is set to zero if no tracing is to be performed for that work unit. The caller (CICS or IMS, for example) propagates the token in a manner similar to the propagation of the service class token.

Transaction trace macros are then used by resource managers to

- determine if tracing can be performed (ITZQUERY)
- initiate the writing of a transaction trace record (ITZEVENT)

Transaction trace writes trace data in a transaction trace data space in the trace address space. If an external writer has been defined, the record is also written to the external writer. IPCS is used to view the transaction trace records.

Transaction Trace Commands

The following commands have been modified for use with transaction trace:

- TRACE TT
- DISPLAY TRACE,TT

For information about using the TRACE or DISPLAY TRACE commands with transaction trace, see *z/OS MVS System Commands*.

The TRACE TT Command

Transaction trace uses the MVS TRACE command with the TT keyword to:

- Start transaction trace.
- Add additional trace filter sets.
- Remove an active trace filter set.
- Stop transaction trace.
- Start a CTRACE external writer.
- Stop a CTRACE external writer.
- Change the transaction trace buffer size.
- Specify a level indicator.
- Specify whether or not latent transactions should be traced.

Starting Transaction Trace

Transaction trace is started when a TRACE TT command is issued with filter information. Following is an example of defining a transaction trace filter set with a userid of TESTERP1 and transaction name of TRAN1.

```
trace tt,user=testerp1,tran=tran1
ITZ002I 'BUFSIZ' IS SET TO 0001M
ITZ001I TRANSACTION TRACE IS NOW ACTIVE WITH FILTER SET 01
```

When multiple filter keywords are specified, as in the above example, a ’logical AND’ is used to determine if the transaction should be traced or not traced.

Adding Additional Trace Filter Sets

Up to five transaction trace filter sets can be concurrently active. They are activated when the TRACE TT command is issued with filter information. The command in the following example defines an additional transaction trace filter set with a userid of DONNA*. The use of an asterisk (*) in the last character position indicates a wildcard is being defined. When determining if a transaction trace token is to be created, any userid with a prefix of DONNA will result in a match.

```
trace tt,user=donna*
ITZ001I TRANSACTION TRACE IS NOW ACTIVE WITH FILTER SET 02
```

If multiple filter sets are specified a ’logical OR’ is used among the filter sets to determine if the transaction should be traced or not traced.

Removing an Active Trace Filter Set

A transaction trace filter set is removed when the OFF=x keyword is used. For example:

```
trace tt,off=2
ITZ016I TRANSACTION TRACE FILTER SET TURNED OFF
```

indicates that the transaction trace filter set 02 has been turned off.

Stopping Transaction Trace

Transaction trace is stopped when the OFF=ALL keyword is used. For example:

```
trace tt,off=all
ITZ007I TRANSACTION TRACE IS NO LONGER ACTIVE.
A DUMP COMMAND MAY BE ISSUED TO DUMP THE TRANSACTION TRACE
DATA SPACE.
```

indicates that transaction tracing has been stopped.

The DUMP command should be used to dump the transaction trace data space. For example, enter

```
DUMP COMM=(TTrace for TRAN=ATM1)
```

followed by

```
R x,DSPNAME='TRACE'.SYSTTRC
```

Starting a CTRACE External Writer

Transaction trace supports the use of an external writer for processing transaction trace records. An external writer can be specified on the initial command that activates transaction trace or specified standalone while transaction trace is active. For example:

```
trace tt,wtr=abcdefg
```

Component trace messages are issued in response to this command.

Stopping a CTRACE External Writer

Transaction trace external writer processing can be stopped with the use of the WTR=OFF keyword. For example:

```
trace tt,wtr=off
```

Component trace messages are issued in response to this command.

Changing the Data Space Size

The transaction trace TTRACE TT command allows the transaction trace data space size to be changed. The data space can be from 16K to 999K or 1M to 32M. For example:

```
trace tt,bufsiz=2m
ITZ002I 'BUFSIZ' IS SET TO 0002M
```

Specifying a Level Indicator

The transaction trace TTRACE TT command allows definition of a level indicator for each filter set.

- 1 pertains to component entry, exit, exceptions, and major events.
- 2 pertains to detail, controlled by component external.

The default is 2. For example:

```
trace tt,bufsiz=2m,user=testerp1,tran=tran1,lvl=01
ITZ002I 'BUFSIZ' IS SET TO 0002M
ITZ001I TRANSACTION TRACE IS NOW ACTIVE WITH FILTER SET 01
```

Tracing Latent Transactions

The transaction trace TTRACE TT command can be used to specify whether or not latent transactions should be traced. The default is to trace latent processing. Consider the following when deciding what to specify:

- The transaction is currently active in the system.
- The transaction is marked for tracing.

- The filter set used to mark the transaction eligible for tracing is no longer active.

An example follows:

```
trace tt,latent=no
ITZ002I 'LATENT' IS SET TO NO
```

DISPLAY TRACE,TT

The TT keyword on the DISPLAY TRACE command is used to determine the status of transaction trace. Do not use the component trace display command to inquire on the status of transaction trace. In addition to displaying information specified on the TRACE TT command, the DISPLAY TRACE,TT response also displays a list of the systems participating in transaction trace sysplex processing.

The following is an example of a DISPLAY TRACE,TT command response:

```
IEE843I 14.47.19  TRACE DISPLAY
SYSTEM STATUS INFORMATION
ST=(ON,0064K,00064K) AS=ON  BR=OFF EX=ON
MT=(ON,024K)
--------------------------------------------------------------
TRANSACTION TRACE STATUS: ON
   BUFSIZ= 0002M     WRITER= *NONE*      LATENT= YES
   01:  TRAN= TRAN1                 USER= TESTERP1
        LVL = 001
   02:  USER=DONNA*               LVL = 002
   SYSTEMS PARTICIPATING IN TT: SYS1     SYS2     SYS3
```

Using IPCS To View Transaction Trace Output

The IPCS subcommand CTRACE COMP(SYSTTRC) is used to view transaction trace records. To obtain a sysplex TTrace stream, use the IPCS MERGE subcommand to format TTrace records from multiple input data sets. Any GTF records imbedded in the TTrace records will be processed without having to specify additional keywords to the above command.

IPCS CTRACE COMP(SYSTTRC) Examples

Following is an example of a short IPCS CTRACE COMP(SYSTTRC) SHORT command response:

```
ctrace comp(systtrc) short
COMPONENT TRACE SHORT FORMAT
COMP(SYSTTRC)
**** 09/23/1999
SYSNAME   MNEMONIC  ENTRY ID  TIME STAMP       DESCRIPTION
-------   --------  --------  ---------------  ----------

SY1       TTCMD     00000002  14:17:20.833847  TRACE TT Command

SY1       TTCMD     00000002  14:18:11.611755  TRACE TT Command

SY1       EVENT     00000003  14:31:55.813125  TRACE EVENT

SY1       EVENT     00000003  14:31:55.899216  TRACE EVENT

SY1       EVENTU    00000005  14:31:56.378480  TRACE EVENT with User
                                                   Data

SY1       EVENTG    00000004  14:31:56.818367  TRACE EVENT with GTF
                                                   Data
```

Following is an example of a IPCS CTRACE COMP(SYSTTRC) LONG command response:

```
ctrace comp(systtrc) full
COMPONENT TRACE FULL FORMAT
COMP(SYSTTRC)
**** 09/23/1999
SYSNAME   MNEMONIC  ENTRY ID  TIME STAMP       DESCRIPTION
-------   --------  --------  ---------------  ----------
SY1       TTCMD     00000002  14:17:20.833847  TRACE TT Command
 CMDID.....0501
 COMMAND...TRACE TT,BUFSIZ=2M,USER=TESTERP1,TRAN=TRAN1,
 LVL=01

SY1       TTCMD     00000002  14:18:11.611755  TRACE TT Command
 CMDID.....0402
 COMMAND...TRACE TT,USER=DONNA*
SY1       EVENT     00000003  14:31:55.813125  TRACE EVENT

 COMPONENT..COMP      EVENTDESC..TTVAPIEA008       CMDID.....0501
 FUNCTION...TEST_ITZEVENT_WITH_FUNCTIONNAME.  TCB...007ED9C8
 ASID..0022
 TRACETOKEN..SY1       B2E447A3  F32E4048  05010100  00000000

SY1       EVENT     00000003  14:31:55.899216  TRACE EVENT

 COMPONENT..COMP      EVENTDESC..TTVAPIEA009       CMDID.....0000
 FUNCTION...................................  TCB...007ED7A8
 ASID..0022
 TRACETOKEN..          40404040  40404040  40404040  40404040
 LATENT workunit traced.

SY1       EVENTU    00000005  14:31:56.378480  TRACE EVENT with User
                                                       Data

 COMPONENT..COMP      EVENTDESC..TTVAPIEA003       CMDID.....0501
 FUNCTION...................................  TCB...007ED148
 ASID..0022
 TRACETOKEN..SY1       B2E447A3  F5D056C1  0105FF00  00000000
  +0000  E3C8C9E2  40C9E240  E3E340C4  C1E3C140 | THIS IS TT DATA  |
  +0010  C6D6D940  C140E3D9  C1D5E2C1  C3E3C9D6 | FOR A TRANSACTIO |
  +0020  D540E3D9  C1C3C540  D9C5C3D6  D9C44BE3 | N TRACE RECORD.T |
  +0030  C8C9E240  C9E240E3  E340C4C1  E3C140C6 | HIS IS TT DATA F |
  +0040  D6D940C1  40E3D9C1  D5E2C1C3  E3C9D6D5 | OR A TRANSACTION |
  +0050  40E3D9C1  C3C540D9  C5C3D6D9  C44B     |  TRACE RECORD.   |

SY1       EVENTG    00000004  14:31:56.818367  TRACE EVENT with GTF
                                                       Data

 COMPONENT..COMP      EVENTDESC..TTVAPIEA004       CMDID.....0402
 FUNCTION...................................  TCB...007ED368
 ASID..0022
 TRACETOKEN..SY1       B2E447A3  F566F584  03030300  00000000
 HEXFORMAT AID FF FID 00 EID  E000
  +0000  E3C8C9E2  40C9E240  C7E3C640  C4C1E3C1 | THIS IS GTF DATA |
  +0010  40C6D6D9  40C140E3  D9C1D5E2  C1C3E3C9 |  FOR A TRANSACTI |
  +0020  D6D540E3  D9C1C3C5  40D9C5C3  D6D9C44B | ON TRACE RECORD. |
  +0030  E3C8C9E2  40C9E240  C7E3C640  C4C1E3C1 | THIS IS GTF DATA |
  +0040  40C6D6D9  40C140E3  D9C1D5E2  C1C3E3C9 |  FOR A TRANSACTI |
  +0050  D6D540E3  D9C1C3C5  40D9C5C3  D6D9C44B | ON TRACE RECORD. |
```

Chapter 13. GETMAIN, FREEMAIN, STORAGE (GFS) Trace

Who's hogging all that virtual storage??!? Let GFS find the culprit!

GFS trace is a diagnostic tool that collects information about the use of the GETMAIN, FREEMAIN, or STORAGE macro. You can use GFS trace to analyze the allocation of virtual storage and identify users of large amounts of virtual storage.

You must use the generalized trace facility (GTF) to get the GFS trace data output.

This chapter contains the following topics:

- "Starting and Stopping GFS Trace"
- "Receiving GFS Trace Data" on page 13-3
- "Formatted GFS Trace Output" on page 13-4
- "Unformatted GFS Trace Output" on page 13-5

Starting and Stopping GFS Trace

The following procedure explains how to request a GFS trace.

1. In the DIAGxx parmlib member, set the VSM TRACE GETFREE parameter to ON and define the GFS trace control data.

   Example: DIAGxx Parmlib Member for Starting GFS Tracing

   The following DIAGxx parmlib member starts GFS trace and limits the trace output to requests to obtain or release virtual storage that is 24 bytes long and resides in address spaces 3, 5, 6, 7, 8 and 9:

   ```
   VSM TRACE GETFREE (ON)
             ASID (3, 5-9)
             LENGTH (24)
             DATA (ALL)
   ```

   Note: If you want the IPCS GTFTRACE output to be formatted, you must include the TYPE and FLAGS data items on the DATA keyword specification of the DIAGxx parmlib member.

   You'll need another DIAGxx parmlib member defined to stop GFS tracing. See 5 on page 13-2.
2. Ask the operator to enter the SET™ DIAG=xx command to activate GFS trace using the definitions in the DIAGxx parmlib member.
3. Start a GTF trace (ask the operator to enter a START *membername* command on the master console). *membername* is the name of the member that contains the source JCL (either a cataloged procedure or a job). Tell the operator to specify a user event identifier X'F65' to trace GTF user trace records.

Example: Starting a GTF trace for GFS data

In the following example, the operator starts GTF tracing with cataloged procedure GTFPROC to get GFS data in the GTF trace output. The contents of cataloged procedure GTFPROC are as follows:

```
//GTF     PROC MEMBER=GTFPROC
//* Starts GTF
//IEFPROC EXEC PGM=AHLGTF,REGION=32M,
//  PARM='MODE=EXT,DEBUG=NO,TIME=YES,BLOK=40K,SD=0K,SA=40K'
//IEFRDER DD DSN=D31POOL.PJREDGTF.TRACE,
//           DISP=SHR,UNIT=3380,VOL=SER=CTDSD1
```

The operator then replies to messages AHL100A with the USRP option. When message AHL101A prompts the operator for the keywords for option USRP, the operator replies with USR=(F65) to get the GFS user trace records in the GTF trace output.

```
START GTFPROC

00 AHL100A SPECIFY TRACE OPTIONS

REPLY 00,TRACE=USRP

01 AHL101A SPECIFY TRACE EVENT KEYWORDS--USR=

REPLY 01,USR=(F65)

02 AHL102A CONTINUE TRACE DEFINITION OR REPLY END

REPLY 02 END

AHL103I TRACE OPTIONS SELECTED--USR=(F65)

03 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U

REPLY 03,U
```

4. To stop the GTF trace, ask the operator to enter a STOP *procname* command on the master console.
5. To stop GFS trace, create a DIAGxx parmlib member with VSM TRACE GETFREE(OFF) and have the operator enter a SET DIAG=xx command.

Example: DIAGxx Parmlib Member for Stopping GFS Tracing

The following DIAGxx parmlib member stops GFS trace:

```
VSM TRACE GETFREE (OFF)
```

References

- See *z/OS MVS Initialization and Tuning Reference* for the syntax of the DIAGxx parmlib member.
- See *z/OS MVS System Commands* for the syntax of the SET and START commands.
- See Chapter 10, “The Generalized Trace Facility (GTF)” on page 10-1 for information about how to specify GTF EIDs.

Receiving GFS Trace Data

GTF places the GFS trace data in a user trace record with event identifier X'F65'. To obtain GFS trace data, do one of the following:

- When GTF writes trace data in a data set, format and print the trace data with the IPCS GTFTRACE subcommand.
- When GTF writes trace data only in the GTF address space, use a dump to see the data. Request the GTF trace data in the dump through the SDATA=TRT dump option.
- Issue the IPCS GTFTRACE subcommand to format and see the trace in an unformatted dump.

Reference

See *z/OS MVS IPCS Commands* for the GTFTRACE subcommand.

Formatted GFS Trace Output

Example: Formatted GFS Trace Output

```
READY
   IPCS NOPARM
IPCS
   DROPD DA('D10JHM1.VSMNEW.GTF')
BLS18206I All records for 1 dump dropped
IPCS
   SETD NOCONFIRM
IPCS
   GTFTRACE DA('D10JHM1.VSMNEW.GTF') USR(F65)
IKJ56650I TIME-03:42:20 PM. CPU-00:00:01 SERVICE-52291 SESSION-00:00:20 JANUARY 22,1998
BLS18122I Initialization in progress for DSNAME('D10JHM1.VSMNEW.GTF')
IKJ56650I TIME-03:42:21 PM. CPU-00:00:01 SERVICE-54062 SESSION-00:00:20 JANUARY 22,1998
   **** GTFTRACE DISPLAY OPTIONS IN EFFECT ****
 USR=SEL

**** GTF DATA COLLECTION OPTIONS IN EFFECT: ****
 USRP option

                 **** GTF TRACING ENVIRONMENT ****
      Release: SP6.0.6   FMID: HBB6606   System name: CMN
      CPU Model: 9672  Version: FF  Serial no. 270067


USRDA F65 ASCB 00FA0800              JOBN GTFJM2
Getmain SVC(120)  Cond=Yes
Loc=(Below,Below)  Bndry=Dblwd
Return address=849CA064  Asid=001A  Jobname=GTFJM2
Subpool=229  Key=0  Asid=001A  Jobname=GTFJM2    TCB=008DCA70 Retcode=0
Storage address=008D6768 Length=10392 X'2898'
  GPR Values
      0-3  00002898  00000000  7FFFC918  0B601E88
      4-7  01FE3240  008FF830  849CA000  00FA0800
      8-11 00000000  00000DE8  049CBFFE  849CA000
     12-15 049CAFFF  0B601A9C  00FE9500  0000E510

               GMT-01/06/1998 21:15:43.111628  LOC-01/06/1998 21:15:43.111628
   .
   .
   .

USRDA F65 ASCB 00FA0800              JOBN GTFJM2
Freemain SVC(120)  Cond=No
Return address=8B2D608A  Asid=001A  Jobname=GTFJM2
Subpool=230  Key=0  Asid=001A  Jobname=GTFJM2    TCB=008DCA70  Retcode=0
Storage address=7F73DFF8  Length=8 X'8'
  GPR Values
      0-3  00000000  7F73DFF8  008D82D8  008D7BC0
      4-7  008D8958  008D6B08  008D85C8  0B335000
      8-11 00000002  00000000  7F73DFF8  008D862C
     12-15 8B2D6044  008D8C98  849D242A  0000E603

               GMT-01/06/1998 21:15:43.111984  LOC-01/06/1998 21:15:43.111984

IPCS
  SETD   CONFIRM
IPCS
  END
READY
END
```

The GETMAIN / FREEMAIN / STORAGE trace produces a second type of record with a slightly different format. Following is an example of this record type:

```
USRDA F65 ASCB 00F4C280           JOBN IYCSCTS6
          Releasing Subpool=230  Key=1  Asid=003E  TCB=008B11E0
          Storage address=7F653E00  Length=512 X'200'
```

This type of record is unique because it does not trace a return address. It is written whenever an individual area of storage is FREEMAINed as part of a subpool FREEMAIN request. There may be many of these records in a row. The last record of the sequence is followed by a record indicating that a subpool FREEMAIN was requested. The return address of the issuer of the subpool FREEMAIN is included on this record.

Unformatted GFS Trace Output

This topic shows unformatted GFS trace output as it would appear in the trace data set where GTF puts the output. You can use this information to write your own formatting or analysis routines.

```
Unformatted GFS Trace Output

Part 1 - This part is in every GFS trace entry.

Offset  Length  Description
---------------------------------------------------------------------
    0      1    Flags
                X'80' - Common storage
                X'40' - Caller's registers are traced
                X'20' - This is a subpool release range entry
                X'10' - Copy of VSWKOWNINFO
---------------------------------------------------------------------
    1      1    Actual subpool after translation
---------------------------------------------------------------------
    2      2    ASID which owns the storage
---------------------------------------------------------------------
    4      4    Address of storage area
---------------------------------------------------------------------
    8      4    Actual length of storage area
---------------------------------------------------------------------
    C      4    Address of TCB
---------------------------------------------------------------------
   10      1    Copy of VSWKSKEY
---------------------------------------------------------------------
   11      1    Copy of VSWKRC
---------------------------------------------------------------------
   12      1    Modification level number  X'01' - HBB6606
                                           X'02' - HBB7703
---------------------------------------------------------------------
   13      1    Reserved
---------------------------------------------------------------------
   14      2    Offset of Part 2
---------------------------------------------------------------------
   16      2    Offset of Part 3
---------------------------------------------------------------------
```

Figure 13-1. Layout of the GFS Trace Output (Part 1 of 2)

Part 2 - This part is in every GRS trace entry except for subpool release range entries.

Offset	Length	Description
0	4	Caller's return address
4	4	Minimum length for a variable request
8	4	Maximum length for a variable request
C	8	Name of job which owns the storage
14	8	Name of job which contained the program which requested the storage
1C	2	ASID which contained the program which requested the storage
1E	1	Copy of VSWKESPL
1F	1	Copy of VSWKSVC
20	1	Copy of VSWKRFLG
21	1	Copy of VSWKPFLG
22	1	Copy of VSWKFLGS
23	1	Copy of VSWKRFLG2
24	4	Copy of VswkRetAddrHigh
28	4	Copy of VswkAR15Value

Part 3 - This part is in the GFS trace record if the caller's registers are traced.

Offset	Length	Description
0	X'40'	Caller's registers 0-15

Figure 13-1. Layout of the GFS Trace Output (Part 2 of 2)

Note: Field names beginning with VSWK are contained within macro IGVVSMWK. Refer to *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)* for more information.

Chapter 14. Recording Logrec Error Records

A treasure trove of gems: symptom strings, error records, VRA listings and much more.

When an error occurs, the system records information about the error in the logrec data set or the logrec log stream. The information provides you with a history of all hardware failures, selected software errors, and selected system conditions. Use the Environmental Record, Editing, and Printing program (EREP):

- To print reports about the system records
- To determine the history of the system
- To learn about a particular error

Collection of Software and Hardware Information

Use the records in the logrec data set or the logrec log stream as additional information when a dump is produced. The information in the records will point you in the right direction while supplying you with symptom data about the failure.

Figure 14-1 shows the error processing for a logrec data set, named SYS1.LOGREC, which is the default name for the logrec data set.

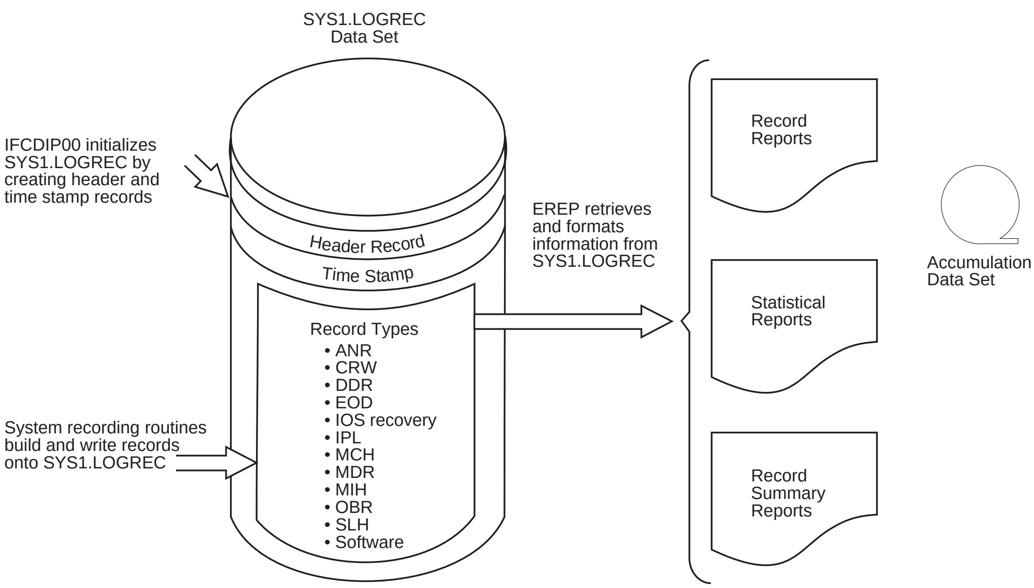

Figure 14-1. Logrec Error Recording Overview

Major Topics

You can set your system up to record errors on either a logrec data set or in a logrec log stream. This chapter tells you what you need to know about each medium before deciding how to collect error records.

To use the logrec data set or a logrec log stream, you need to know how to initialize each of them, how to record system events on each of them, how to collect the data when it is available, and how to interpret the output through EREP. This chapter describes each of these tasks:

- “Choosing the Correct Logrec Recording Medium”
- “Initializing and Reinitializing the Logrec Data Set”
- “Defining a Logrec Log Stream” on page 14-4
- “Error Recording Contents” on page 14-6
- “Obtaining Information from the Logrec Data Set” on page 14-10
- “Obtaining Records from the Logrec Log Stream” on page 14-12
- “Obtaining Information from the Logrec Recording Control Buffer” on page 14-18
- “Interpreting Software Records” on page 14-19

Choosing the Correct Logrec Recording Medium

You can choose where the system will record logrec error records. When a system is not in a sysplex, an installation can use a logrec data set, associated with an individual system, to record error records. An installation can choose to continue this type of recording by initializing the logrec data set before IPLing the system that will use it.

In a sysplex, however, because each system requires its own logrec data set, you might need to look at each logrec data set when an error occurs.

To eliminate the problem of having to manage up to 32 logrec data sets, an installation can choose to define one coupling facility logrec log stream. Using a coupling facility logrec log stream eliminates the following:
- Running IFCDIP00 to initialize multiple logrec data sets
- Handling full or emergency data set conditions
- Scheduling the daily offload of logrec data sets
- Concatenating multiple history data sets
- Archiving logrec records

References
- See “Initializing and Reinitializing the Logrec Data Set” if you want to initialize a logrec data set for your system.
- See “Defining a Logrec Log Stream” on page 14-4 if you want to define a logrec log stream for your installation.

Initializing and Reinitializing the Logrec Data Set

You must initialize the logrec data set before IPLing the system that will use it.

You reinitialize the logrec data set when an uncorrectable error occurs. You clear the logrec data set when it is full or near full.

To initialize or reinitialize the logrec data set, use the service aid program IFCDIP00. To clear a full logrec data set, use EREP. IFCDIP00 creates a header record and a time stamp record for the logrec data set.

Attention: The logrec data set is an unmovable data set. If you attempt to move it after IPL using a program, such as a defragmentation program, your system will experience difficulty both reading from and writing to the data set.

Initializing the Logrec Data Set

If the logrec data set does not exist, you must first allocate it and then initialize it. (Whenever you allocate or reallocate the logrec data set, the newly allocated data set will not be used until you initialize it and IPL the system on which it is to be used.)

Following is an example of a job that scratches and uncatalogs an existing logrec data set and allocates, catalogs, and initializes a new one. (If you do not currently have a logrec data set, start with the second step of the job.)

Example: Changing the Space Allocation

Use the JCL statements below to do the following:

- Rename the logrec data set. For example, rename SYS1.LOGREC to SYS1.LOGREC.OLD.
- Allocate and initialize a new logrec data set with new space specifications using IFCDIP00.

```
//KATHYLR JOB (9999),'CREATE NEW LOGREC DS',CLASS=A,MSGCLASS=X,
//        MSGLEVEL=(1,1),NOTIFY=KATHY
//*-----------------------------------------------------------------
//*  RENAME THE CURRENT LOGREC DATASET
//*  UNCATLG SYS1.LOGREC SO THE NEW LOGREC CAN BE ALLOCATED ON
//*  ANOTHER VOLUME, IF DESIRED
//*-----------------------------------------------------------------
//RENAME    EXEC PGM=IEHPROGM
//M43RES    DD VOL=SER=M43RES,UNIT=3390,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
         RENAME DSNAME=SYS1.LOGREC,VOL=3390=M43RES,                    X
               NEWNAME=SYS1.LOGREC.OLD
         UNCATLG DSNAME=SYS1.LOGREC
/*
//*-----------------------------------------------------------------
//*  CREATE THE NEW LOGREC DATASET AND INITIALIZE IT
//*-----------------------------------------------------------------
//IFCDIP00 EXEC    PGM=IFCDIP00,COND=(0,LT)
//SERERDS  DD  DSN=SYS1.LOGREC,DISP=(,CATLG),
//         VOL=SER=M43RES,UNIT=SYSDA,SPACE=(CYL,3,,CONTIG)
/*
                                                         //
```

Note: If you run the preceding JCL and an error occurs after the logrec data set has been scratched but before it has been reallocated, you will be unable to IPL your system using this logrec data set.

To solve this problem, do one of the following:

- Use the DFDSS stand-alone restore program to restore your old logrec data set.
- Run the reallocate job on the data set while running under another system.

References

- See *z/OS DFSMShsm Storage Administration Reference* for information about the DFDSS stand-alone restore program.
- See *z/OS DFSMSdss Storage Administration Reference* for information about the DFDSS stand-alone restore program.

Reinitializing the Logrec Data Set

You need to reinitialize the logrec data set either when the data set is full or when an uncorrectable error occurs.

If the data set is full, use EREP to record the data in a history data set and reinitialize logrec.

In the case of an error, invoke IFCDIP00 with JCL statements to reinitialize your existing logrec data set. IFCDIP00 resets the logrec data set header record field to indicate that the entire data set can be used and clears the time stamp record to hexadecimal zeros.

For information on using EREP, see the *EREP User's Guide*.

Following is an example of using the IFCDIP00 service aid to reinitialize the logrec data set:

Example: Reinitializing the Logrec Data Set

Use the following JCL statements:

```
//INSERLOG  JOB
//STEP1     EXEC  PGM=IFCDIP00
//SERERDS   DD    DSNAME=SYS1.LOGREC,UNIT=3380,
//     VOL=SER=111111,DISP=(OLD,KEEP)
```

The JOB statement initiates the job; the job name INSERLOG has no significance.

The EXEC statement specifies the program name (PGM=IFCDIP00).

The SERERDS DD statement specifies the reinitialized logrec data set (in this case SYS1.LOGREC), which must be on a permanently mounted volume (VOL=SER=111111 in this example); the DDNAME must be SERERDS.

Defining a Logrec Log Stream

Before defining a logrec log stream, note that IBM recommends that you IPL with a logrec data set initialized by IFCDIP00. If you do not IPL with a data set, you cannot change the logrec recording medium from LOGSTREAM to DATASET using the SETLOGRC command.

To use the logrec log stream, you must first prepare your installation to use system logger functions. IBM recommends that you use a coupling facility log stream for LOGREC so that you can merge data from multiple systems in a sysplex.

To obtain logrec records for a single system sysplex, you can also use a DASD-only log stream, which is single system in scope. Note that this is not recommended for a multi-system sysplex, because you can only have one logrec log stream per sysplex. This means that if you make your logrec log stream DASD-only, only one system will be able to access it. See the system logger chapter of *z/OS MVS Setting Up a Sysplex* for information on DASD-only log streams.

Reference

See *z/OS MVS Setting Up a Sysplex* for more information.

The following steps describe how to use a coupling facility logrec log stream in place of a logrec data set:

1. Define a log stream named SYSPLEX.LOGREC.ALLRECS using the system logger log stream definition utility, IXCMIAPU.

Example: Sample JCL of Using IXCMIAPU

IFBLSJCL is available in SYS1.SAMPLIB as an example of using the administrative data utility, IXCMIAPU, to define the coupling facility logrec log stream to a sysplex.

```
//IFBLSJCL JOB
//* Member Name:  IFBLSJCL                                            *
//*  Descriptive Name:                                                *
//*   Sample JCL to provide an example of using the System Logger     *
//*   utility to define the Logrec log stream to a sysplex.           *
//* Function:                                                         *
//*   This JCL sample provides an example of running the System       *
//*   Logger utility (IXCMIAPU) to define the Logrec log stream       *
//*   in the logger inventory.                                        *
//*                                                                   *
//*   Note that the MAXBUFSIZE parameter must have at least 4068      *
//*   specified, or Logrec will not be able to write to the Log       *
//*   stream.                                                         *
//*                                                                   *
//*   The Logrec log stream name must be specified as                 *
//*   SYSPLEX.LOGREC.ALLRECS.                                         *
//*                                                                   *
//* Suggested Modifications:                                          *
//*   Provide the specifications that are relevant for your           *
//*   installation on the SYSIN DATA TYPE(LOGR) definition.           *
//*   For example, the following parameters define the log stream     *
//*   data set attributes:                                            *
//*                                                                   *
//*    LS_DATACLAS(data class)        - Name of data class            *
//*    LS_MGMTCLAS(management class)  - Name of management class      *
//*    LS_STORCLAS(storage class)     - Name of storage class         *
//*                                                                   *
//* Distribution Library:  ASAMPLIB                                   *
//*                                                                   *
//DEFINE   EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
 DATA TYPE (LOGR)
    DEFINE STRUCTURE NAME(LOGRECSTRUCTURE)
           LOGSNUM(1)
           AVGBUFSIZE(4068)
           MAXBUFSIZE(4068)
    DEFINE LOGSTREAM NAME(SYSPLEX.LOGREC.ALLRECS)
           STRUCTNAME(LOGRECSTRUCTURE)
/*
```

Note: MAXBUFSIZE must be at least 4068 because logrec writes records in one page blocks. Specify SMS storage group, storage, data and management classes such that when one data set is full, another is allocated. Allocate as much space as is allocated for all the logrec data sets on the systems in the sysplex before migrating to a logrec log stream.

The most effective way to manage all logrec records is to specify the automatic migration of log data sets to HSM. This automatic migration eliminates the need to create and maintain archival history data sets, with one exception. If the log stream data set directory is full, you can, using *SUBSYS-options2* of the LOGR subsystem, copy data from a log stream to a history data set and then delete the copied data from the log stream.

2. Either specify LOGREC=LOGSTREAM in the IEASYSxx parmlib member or, after IPLing with LOGREC=dsname, use the SETLOGRC command to change the logrec recording medium to a logrec log stream. Any records written to any logrec data sets before changing to a logrec log stream must be read by a separate EREP job. If you IPL the system with LOGREC=LOGSTREAM, you cannot use the SETLOGRC command to change the logrec recording medium to a logrec data set.
3. Change the EREP job stream as follows:
   - Change the SERLOG DD DSN=SYS1.LOGREC statement associated with a logrec data set to an ACCIN DD DSN=SYSPLEX.LOGREC.ALLRECS statement, with corresponding SUBSYS parameters, to associate EREP with the logrec log stream. The SUBSYS parameters are described in “Obtaining Records from the Logrec Log Stream” on page 14-12.
   - Identify the input as a history data set. Leave the output to a history data set as currently recommended, because all subsequent steps should already use the history data set as input.

     Note: Using a logrec log stream as input for multiple steps is not recommended because each subsequent step processes more records than the prior, causing numbers and data in successive reports not to match.
   - Subsequent EREP report steps that normally process history data sets no longer need to concatenate one history data set per system.

References

- See *z/OS MVS Setting Up a Sysplex* for information about preparing an installation to use system logger functions.
- See *EREP User’s Guide* for more information about running an EREP job to obtain a history data set.
- See *z/OS MVS System Commands* for more information about the SETLOGRC command.
- See *z/OS MVS Initialization and Tuning Reference* for more information about the IEASYSxx parmlib member.

Error Recording Contents

The system creates records for every hardware or software failure and system condition that occurs and stores these records in the logrec data set or the logrec log stream. The records can contain two types of data that document failures and system conditions:

- **Error statistics**, which include the number of times that channels, machine models, and I/O devices have failed

- **Environmental data**, which include time and circumstances for each failure or system condition

Note: A programmer can also build symptom records using the SYMRBLD macro and have those records written into the logrec data set or the logrec log stream using the SYMREC macro.

Reference

See *z/OS MVS Programming: Assembler Services Reference IAR-XCT* for information about the macros.

Each record is recorded in hexadecimal format as an undefined length record. Each record provides:
- Relevant system information at the time of the failure
- Device hardware status at the time of the failure
- Results of any device/control unit recovery attempt
- Results of any software system recovery attempt
- Statistical data

When taken as a whole, these records create a history of the system, which begins early in system initialization and ends when the system stops. These records contain:
- **Full Abend History**: The system writes a logrec record for every abend, regardless of whether the dump is requested or suppressed. The logrec data set or the logrec log stream contains a full record of abnormal ends.
- **System Initialization Errors**: The system writes errors during system initialization, before other diagnostic services are completely functioning.
- **Lost Record Counts**: The system writes a logrec record to summarize lost error records. Sometimes hardware-detected or software-detected errors occur close together. When errors are too close together, the system cannot write an individual record for each error; instead, the system counts the errors and writes a summary record.

These sections describe what is in the logrec data set:
- “Logrec Data Set Header Record”
- “Logrec Data Set Time Stamp Record” on page 14-8

This section describes what is in the logrec data set or the logrec log stream:
- “Types of Logrec Error Records” on page 14-8

Reference

See *z/OS MVS Diagnosis: Reference* for the format of the header record, time stamp record, and logrec error records.

Logrec Data Set Header Record

IFCDIP00 creates a header record on the logrec data set. The logrec data set header record includes:
- Information that the system recording routines can use to determine where to write new record entries onto the logrec data set
- Information that EREP can use to find existing record entries on the logrec data set. This information is valuable when you run an EREP report to find a particular error.

- Information that the system recording routines can use to issue a warning message when the logrec data set is 90% full.

Note: The logrec log stream does not have a header record generated.

Logrec Data Set Time Stamp Record

IFCDIP00 creates a time stamp record on the logrec data set in the first record space following the header record. The time stamp record provides current date and time information for the IPL record. This allows you to measure the approximate time interval, recorded in the IPL records, between the ending and reinitialization of the operating system.

At preset time intervals, the system obtains the current date and time and writes this information on the time stamp record, overlaying the previous date and time.

During a subsequent initialization of the system, the system obtains the date and time from the time stamp record and adds it to the IPL record.

If IFCDIP00 is used to reinitialize the logrec data set, the information in the time stamp record is overlaid with hexadecimal zeros until the system writes the current date and time.

Note: The logrec log stream does not have a time stamp record generated.

Types of Logrec Error Records

When the logrec data set or the logrec log stream is initialized, the system begins recording events. The system records the following types of error records, containing device-dependent or incident-dependent information:

- **Asynchronous notification records (ANR):**
  - External timer reference (ETR) records for information related to Sysplex Timer® incidents.
  - Direct access storage device (DASD)-service information message (SIM) records for information concerning servicing needs.
  - Link maintenance information (LMI) records for information for a particular link incident.
- **Channel report word (CRW) records** for:
  - Channel path error
  - Subchannel error
  - Configuration alert error
  - Monitoring facility error
- **Dynamic device reconfiguration (DDR) records** for:
  - Operator and system swaps between direct access and magnetic tape devices
  - Operator swaps on unit record devices
- **End-of-day (EOD) records** for information related to end-of-day and system ending conditions whenever the RDE option has been included in the system.
- **Input/output supervisor (IOS) records** for information related to IOS recovery actions.
  - Dynamic pathing services validation (DPSV) records for recovery actions.
- **Initial program load (IPL) records** for information related to system initializations whenever the RDE option has been included in the system.
- **Machine check handler (MCH) records** for:

  - Central processor failure
  - Storage failure
  - Storage key failure
  - Timer failure
- **Miscellaneous data (MDR) records** for:
  - Buffer overflow and device failures on buffered log devices
  - Demounts on DASD with buffered logs
  - Demounts by the DFDSS program between DASD having buffered logs and removable disk packs
  - Device failures on teleprocessing devices connected to an IBM communication controller
  - Statistical recording by EREP on DASD with buffered logs
- **Missing interruption handler (MIH) records** for:
  - Missing I/O interruptions
  - Specified time intervals
  - Recovery actions required
  - Recovery actions performed
- **Outboard (OBR) records** for:
  - Counter overflow statistics and device failures on devices supported by the teleprocessing access methods
  - End-of-day (EOD) requests
  - Paging I/O errors
  - Permanent channel and I/O device failures
  - Statistic counter overflow
  - Temporary or intermittent I/O device failures
  - Demounts on IBM magnetic tape drives
  - Devices that have their own diagnostic buffers
  - Statistical recording by EREP on DASD with buffered logs
- **Subchannel logout handler (SLH) records** for channel errors.
- **Software records**, including:
  - Machine checks (hardware-detected hardware errors, such as software recovery attempts for hard machine failures)
  - Program checks (hardware-detected software errors)
  - Restart errors (operator-detected errors)
  - Lost record errors (count of the records that did not fit in the buffer to be written to the logrec data set)
  - Software-detected errors, such as:
    - Abnormal ends, which are also called *abends*; reported in software records or erroneous supervisor call (SVC) instructions. These are known as SDWA-type software records.
    - Errors that are not abnormal ends; reported in symptom records.
    - Errors generated by application programs or system components; reported in symptom records.

As you can see, the system records a comprehensive list of error records that can help you when you need to diagnose a system failure.

Obtaining Information from the Logrec Data Set

You can obtain the information recorded in the logrec data set using EREP, which formats error records.

EREP can perform the following functions:
- Create an accumulation data set from the logrec data set
- Clear the logrec data set
- Copy an input accumulation data set to an output accumulation data set
- Merge data from an accumulation data set and the logrec data set
- Print a detailed description of selected hardware and software error records
- Summarize and print statistics for device failures

EREP places the information from the logrec data set into reports. Using JCL, you determine the type of report you want EREP to produce.

Using EREP

EREP presents information from the logrec software error records in five reports.

Detail Edit Report for an Abend

The system obtains most of the information for an abend logrec error record from the system diagnostic work area (SDWA). The report contents are:

- Record header: report type (SOFTWARE RECORD), system, job name, error identifier (ERRORID), date, and time
- Search argument abstract
- Serviceability information
- Time of error information
- Status information from the request block
- Recovery environment
- Recovery routine action
- Hexadecimal dump of the SDWA, including the variable recording area (VRA)

Example: Printing a Detail Edit Report

The following example shows how to generate detail edits and summaries of all software and operational records:

```
//STEP7    EXEC  PGM=IFCEREP1,PARM='CARD'
//ACCIN      DD  DSN=EHISTORY,DISP=(OLD,PASS)
//DIRECTWK  DD  UNIT=SYSDA,
//               SPACE=(CYL,5,,CONTIG)
//EREPPT     DD  SYSOUT=A,DCB=BLKSIZE=133
//TOURIST    DD  SYSOUT=A,DCB=BLKSIZE=133
//SYSIN      DD  DSN=EREP.PARMS(STEP7),
//               DISP=(OLD,PASS)
PRINT=PS
TYPE=SIE
HIST
ACC=N
ENDPARM
```

Detail Edit Report for a Symptom Record

The system obtains most of the information for a non-abend logrec error record from the symptom record identified in the SYMREC macro. A programmer can build the symptom record using the SYMRBLD macro. The report contents are:

- Record reader: report type (SYMPTOM RECORD), system, date, and time
- Search argument abstract
- System environment
- Component information
- Primary and secondary symptom strings
- Free-format component information
- Hexadecimal dump of the symptom record

System Summary Report

The report summarizes errors for each of your installation's principle parts, or subsystems: processors, channels, subchannels, storage, operating system control programs, and I/O subsystems. The report contents are:

- Record header: report type (SYSTEM SUMMARY), system, date, time
- Total errors and errors for each processor for the following types of errors:
  - IPL
  - Machine check
  - Program error
  - End of day
- Identifications for processors in the report

Event History Report

The report shows the error history: the frequency, order, and pattern of errors. The report contents are:

- Record header: report type (EVENT HISTORY)
- Abstracts for abend and non-abend logrec error records in chronological order
- Totals of the types of logrec error records for the system and for each processor

Example: Printing an Event History Report

The following JCL defines a two-step job. The first step prints an event history report for all logrec data set records. The second step formats each software, IPL, and EOD record individually. The event history report is printed as a result of the EVENT=Y parameter on the EXEC statement of the first step. It can be a very useful tool to the problem solver because it prints the records in the same sequence they were recorded and therefore shows an interaction between hardware error records and software error records.

```
//EREP JOB MSGLEVEL=1
//EREPA EXEC PGM=IFCEREP1,PARM='EVENT=Y,ACC=N',
//   REGION=128K
//SERLOG DD DSN=SYS1.LOGREC,DISP=SHR
//TOURIST DD SYSOUT=A
//EREPPT DD SYSOUT=A,DCB=BLKSIZE=133
//EREPB EXEC PGM=IFCEREP1,PARM='TYPE=SIE,PRINT=PS,ACC=',
//   REGION=128K
//SERLOG DD DSN=SYS1.LOGREC,DISP=SHR
//TOURIST DD SYSOUT=A
//EREPPT DD SYSOUT=A,DCB=BLKSIZE=133
/*
```

Detail Summary Report

The report summarizes information about data in logrec error records. The report contents are:

- Record header: report type being summarized

- Summary information and counts

Example: Printing a Detail Summary Report

The following example shows how to generate detail summaries of all I/O errors:

```
//STEP6   EXEC  PGM=IFCEREP1,PARM='CARD'
//ACCIN    DD DSN=EHISTORY,DISP=(OLD,PASS)
//DIRECTWK DD UNIT=SYSDA,
//            SPACE=(CYL,5,,CONTIG)
//EREPPT   DD SYSOUT=A,DCB=BLKSIZE=133
//TOURIST  DD SYSOUT=A,DCB=BLKSIZE=133
//SYSIN     DD  DSN=EREP.PARMS(STEP6),
//              DISP=(OLD,PASS)
//          DD  DSN=EREP.CONTROLS,
//              DISP=(OLD,PASS)
PRINT=SU
TYPE=DOTH
DEV=(N34XX,N3704,N3705,N3720,N3725,N3745)
HIST
ACC=N
ENDPARM
```

Obtaining Records from the Logrec Log Stream

You can access records in the logrec log stream by either:

- Writing a program using IXGCONN and IXGBRWSE services, see "Using System Logger Services to Obtain Records from the Logrec Log Stream".
- Using EREP. see "Using EREP to Obtain Records from the Logrec Log Stream".

Using System Logger Services to Obtain Records from the Logrec Log Stream

You can obtain records from the logrec log stream by writing a program that uses the IXGCONN and IXGBRWSE system logger services to return log data. The data returned by the IXGBRWSE service for the logrec log stream is mapped by the IFBLOGLB data area. (See *z/OS MVS Programming: Assembler Services Guide* for information on using system logger services.)

Note that the logrec log stream output from the IXGBRWSE service contains an individual log stream record. However, the log stream record actually contains a group of records. The logrec log stream record is mapped by the IFBLOGLB mapping macro. See *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)* for information on the IFBLOGLB mapping macro.

Using EREP to Obtain Records from the Logrec Log Stream

You can use EREP to access the records in the logrec log stream for each system. The log stream subsystem allows existing programs to access error records from a log stream in the same way records were accessed from a logrec data set. See *z/OS MVS Programming: Assembler Services Guide* for information about using and starting the log stream subsystem.

Log Stream Subsystem Data Set JCL Specification

Use the SUBSYS parameter to invoke the LOGR subsystem to access log stream data:

```
//ddname   DD  DSNAME=log.stream.name,
//             SUBSYS=(LOGR[,exit_routine_name][,'SUBSYS-options1'][,'SUBSYS-options2'])
where:
SUBSYS-options1:
[FROM={({[yyyy/ddd][,hh:mm[:ss]]})  | OLDEST}]
[TO={({[yyyy/ddd][,hh:mm[:ss]]})  | YOUNGEST}]
[,DURATION=(nnnn,HOURS)]
[,VIEW={ACTIVE|ALL|INACTIVE}]
[,GMT|LOCAL]
SUBSYS-options2:
defined by the log stream owner
```

Figure 14-2. Log Stream SUBSYS Data Set Specification

Note: Quotation marks around keywords are required when parentheses, commas, equal signs or blank characters are used within the SUBSYS keyword.

Other DD keywords will be validated, if specified, but will be ignored in the LOGR subsystem processing.

DSNAME=*log.stream.name*
Specifies the name of the log stream to read. The name can be 1 to 26 characters in a data set name format.

SUBSYS=(LOGR[,*exit_routine_name***][,***'SUBSYS-options1'***][,***'SUBSYS-options2'***])**
Specifies that processing of this DD is to be handled by the LOGR subsystem.

The *exit_routine_name* is the second positional parameter and specifies the name of the exit routine to receive control from the LOGR subsystem. If the *exit_routine_name* parameter is not specified (null), the default log stream subsystem exit routine, IXGSEXIT, will be used. To access records from the logrec log stream, specify IFBSEXIT.

SUBSYS-options1
Specifies options meaningful to all exit routines. See the documentation for a specific log stream exit for exceptions to these common options. The keywords are:

FROM=*starting_time*
Indicates the starting time of the first log stream block to be processed based on the log stream view that the VIEW keyword specifies. The first block will be the one with a time stamp later than or equal to the specified time.

OLDEST
Indicates the first block read will be the oldest block on the log stream. OLDEST is the default.

yyyy/ddd
Specifies the start date. If the date is omitted, the current date is assumed.

yyyy is a four-digit year number and *ddd* is a three-digit day number from 001 through 366 (366 is valid only on leap years). For example, code February 20, 2000 as 2000/051, and code December 31, 1996 as 1996/366.

hh:mm[:ss]
Specifies the start time. If the time is omitted, the first block written after midnight will be used.

hh is a two digit hour number from 00 to 23, *mm* is a two digit minute number from 00 to 59, and *ss* is a two digit second number from 00 to 59. The seconds field and associated : delimiter can be omitted if not required by the log stream owner.

The FROM keyword is mutually exclusive with the DURATION keyword.

TO=*ending_time*
Indicates the ending time of the last log stream block to be processed based on the log stream view that the VIEW keyword specifies. The last block will be the one with a time stamp earlier than or equal to the specified time.

YOUNGEST
Indicates the last block read will be the youngest block on the log stream at the time the allocation for the DD occurs. YOUNGEST is the default.

yyyy/ddd
Specifies the end date. If the date is omitted, the current date is assumed.

yyyy is a four-digit year number and *ddd* is a three-digit day number from 001 through 366 (366 is valid only on leap years). For example, code March 7, 2001 as 2001/066, and code November 12, 2000 as 2000/317.

hh:mm[:ss]
Specifies the end time. If the time is omitted, the last block written before midnight will be used. If the end date is the same as the current day, then the youngest block on the log stream at the time the allocation for the DD occurs will be used.

hh is a two digit hour number from 00 to 23, *mm* is a two digit minute number from 00 to 59, and *ss* is a two digit second number from 00 to 59. The seconds field and associated : delimiter can be omitted if not required by the log stream owner.

The TO keyword is mutually exclusive with the DURATION keyword.

Note: If the value specified for the FROM keyword is greater than the value specified for the TO keyword, the system ends the jobstep with a JCL error.

DURATION=(*nnnn***,HOURS)**
Specifies which blocks are to be processed. Each *n* is a numeric from 0 to 9. Specifying (nnnn,HOURS) requests the blocks for the last *nnnn* hours up to the youngest block be processed based on the log stream view that the VIEW keyword specifies. The last *nnnn* hours are calculated from the current time of the allocation for the DD.

The first block will be the one with a time stamp greater than or equal to the calculated start time. The last block read will be the youngest block on the log stream at the time the allocation for the DD occurs.

The DURATION keyword is mutually exclusive with the TO and the FROM keywords.

VIEW=ACTIVE|ALL|INACTIVE
Specifies the view or portion of log data to be used to obtain records from the log stream.

System logger maintains two kinds of log stream data in a log stream: an active portion and an inactive portion. The active portion of the log stream is the log data that the log stream owner has not logically deleted through an IXGDELET request. The inactive portion of the log stream is the log data that the log stream owner has logically deleted but that has not yet been physically deleted from the log stream because the retention period (RETPD) specified for the log stream has not yet expired.

The VIEW option designates the portion(s) of the log stream to be used to obtain log data from the log stream, in addition to applying the other parameters.

Because the other parameters also apply, the combination of the FROM, TO, or DURATION parameters and the VIEW parameter might mean that the log stream subsystem exit returns to log data or only a portion of the intended log data. For example, if FROM=starting_time and VIEW=INACTIVE are both specified, and the starting_time is later (younger) than the log data in the inactive portion of the log stream, then there is no log data to meet the access criteria. In the same way, if TO=ending_time and VIEW=ACTIVE are both specified, and the ending_time is earlier (older) than the log data in the active portion of the log stream, then there is no log data to meet the access criteria.

ACTIVE
The view of the log stream is to include only active log data, in addition to applying the other log stream access parameters. ACTIVE is the default.

ALL
The view of the log stream is to include both active and inactive log data, in addition to applying the other log stream access parameters.

INACTIVE
The view of the log stream is to include only the inactive log data, in addition to applying the other log stream access parameters.

GMT|LOCAL
Specifies whether the time is local time (based on the time zone offset at the time the log was written) or GMT time. GMT is the default.

Along with the above general parameters that can be specified for a log stream subsystem data set, system logger provides additional parameters in the *SUBSYS-options2* specifications. The following values can be coded for a logrec log stream:

SUBSYS-options2
Specifies unique exit routine options. Refer to information provided by the specific log stream owner concerning these parameters.

LASTRUN
Indicates that the starting point of the records to be read from the logrec log stream will be from the last record read by a previous use of an application that used LASTRUN. The end point of the records will be to the youngest block in the logrec log stream.

LASTRUN is mutually exclusive with the FROM, TO and DURATION keywords in *SUBSYS-options1* and with DELETE from *SUBSYS-options2*.

DELETE
Indicates that log stream records are to be deleted from the logrec log stream. The log stream itself is not deleted and remains available for use.

If the logrec log stream has been opened in the job step, all records up to but not including the last complete block read by the program will be deleted from the logrec log stream.

If the logrec log stream has not been opened in the job step, all records prior to the time indicated on the TO keyword will not be deleted from the logrec log stream.

DELETE is mutually exclusive with the FROM and DURATION keywords in *SUBSYS-options1* and the LASTRUN and SYSTEM keywords from *SUBSYS-options2*.

DEVICESTATS
Requests that the device statistics kept on the system where this job is running are to be recorded in the logrec log stream before any records are read.

SYSTEM=*system name*
Indicates that only records originating from the specified *system name* are to be returned to the application reading the logrec log stream.

The *system name* value should match the name specified in the SYSNAME parameter of the IEASYSxx parmlib member.

SYSTEM is mutually exclusive with the DELETE keyword from SUBSYS-options2.

Time of Day Considerations

When using the SUBSYS DD statement for LOGR, handle the time of day filtering carefully. The SUBSYS parameter does not accept a stop time of 24:00, but the EREP parameters do accept 24:00 as a stop time.

If it is necessary to write JCL and EREP control statements, you might have to request filtering through both the SUBSYS DD statement and the EREP parameters:

- SUBSYS parameters use blocks of records, and filtering of these blocks is done using time stamps assigned after each logical record enclosed in a block has been assigned its own time stamp.

Example: Using SUBSYS Parameters

To select logrec log stream records that were produced between 05:00 on June 1st, 1997, and the end of that day, code the following:

```
//ACCIN      DD DSN=SYSPLEX.LOGREC.ALLRECS,DISP=SHR,
//              DCB=(RECFM=VB,BLKSIZE=4000),
//              SUBSYS=(LOGR,IFBSEXIT,
//              'FROM=(1997/152,05:00),TO=(1997/153,23:59),GMT')
```

- EREP parameters use logrec logical records. When you use the TIME parameter with EREP, you are specifying a range of hours and minutes of interest on each day selected.

Example: Using EREP Parameters

To select logrec records that were produced between 05:00 on June 1st, 1997, and the end of that day, code the following:

```
DATE=(97152-97152),TIME=(0500-2400)
```

Note that coding the following would be an error:

```
DATE=(97152-97153),TIME=(0500-0000)
```

Example: Creating a History Data Set

Use the following JCL to create a history data set from log data recorded on the logrec log stream.

In this example, DEVICESTATS requests device statistics and the records are to be recorded in the log stream. Records are read from the last block that was processed on the previous submission of a ″LASTRUN″ EREP job up to the youngest block in the log stream. The first time a job with the ″LASTRUN″ option is run, the records are read from the oldest block in the log stream. read from the LASTRUN application.

```
//EREPDALY EXEC  PGM=IFCEREP1,PARM=('HIST,ACC=Y,SYSUM')
//ACCIN    DD DSN=SYSPLEX.LOGREC.ALLRECS,
//            SUBSYS=(LOGR,IFBSEXIT,,'DEVICESTATS,LASTRUN'),
//            DCB=(RECFM=VB,BLKSIZE=4000)
//ACCDEV   DD DSN=EREP.HISTORY,
//            DISP=(NEW,CATLG),
//            DCB=(RECFM=VB,BLKSIZE=4000),
//            UNIT=SYSDA,SPACE=(CYL,(25,5))
//SERLOG   DD DUMMY
//DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,15,,CONTIG)
//TOURIST  DD SYSOUT=A,DCB=BLKSIZE=133
//EREPPT   DD SYSOUT=A,DCB=BLKSIZE=133
//SYSABEND DD SYSOUT=A
//SYSIN    DD DUMMY
/*
```

Example: Producing an Event History

Use the following JCL to produce an event history report from records on the logrec log stream. By not specifying the FROM or TO keywords, the default is FROM=OLDEST and TO=YOUNGEST, indicating processing should include records from the beginning of the log stream to the end of the log stream. By specifying a print data set, EREPPT, the report can be browsed online for an overview of significant activity.

```
//EREPNOW  EXEC  PGM=IFCEREP1,REGION=4M,
//               PARM='CARD'
//ACCIN      DD  DSN=SYSPLEX.LOGREC.ALLRECS,
//               DISP=SHR,
//               SUBSYS=(LOGR,IFBSEXIT,,)
//DIRECTWK   DD  UNIT=SYSDA,SPACE=(CYL,5,,CONTIG)
//EREPPT     DD DSN=EREP.EVENT,DISP=(NEW,CATLG),
//            DCB=BLKSIZE=133,
//            UNIT=SYSDA,SPACE=(CYL,(25,5))
//TOURIST    DD SYSOUT=A,DCB=BLKSIZE=133
//SYSABEND   DD SYSOUT=A
//SYSIN      DD *
             EVENT
             HIST
             ACC=N
             TYPE=ACDEHIMOSX
             ENDPARM
/*
```

When reading records by date and time, you can provide both EREP and SUBSYS parameters. EREP selects records from those passed to it from the SUBSYS parameters.

Obtaining Information from the Logrec Recording Control Buffer

When the system writes a dump, the dump includes the records in the logrec buffer in storage; the buffer records have been either written to the logrec data set or are queued to be written to the logrec data set.

When you begin to diagnose a dump for a system problem, you can use IPCS to view the system records in the logrec recording control buffer.

The logrec recording control buffer is one of the most important areas to be used when analyzing problems in MVS. This buffer serves as the interim storage location for hardware and software error records that are queued to be written to the logrec data set. The buffer is significant because of the error history it contains. Also, any records in the buffer that have *not* reached the logrec data set are almost certainly related to the problem you are trying to solve.

Formatting the Logrec Buffer

To format the logrec buffer, use the IPCS subcommand VERBEXIT LOGDATA. The entries that are still in the buffer will be formatted in the same way as entries that are printed in the EREP detail edit report.

Finding the Logrec and WTO Recording Control Buffers

There are two recording control buffers (RCB) in the SQA. The system uses one buffer for logrec messages, and the other for WTO messages. The CVT+X'16C'

(CVTRBCB) points to the recording buffers control block (RBCB). The RBCB contains the following information about the two recording control blocks (which are also referred to as RCBs or buffers):

For the logrec RCB:
- RBCB+X'10' (RBCBLRCB) points to the logrec buffer.
- RBCB+X'14' (RBCBLLEN) contains the length of the logrec buffer.

For the WTO RCB:
- RBCB+X'18' (RBCBWRCB) points to the WTO buffer.
- RBCB+X'1C' (RBCBWLEN) contains the length of the WTO buffer.

The logrec and WTO recording control buffers reside in fetch-protected SQA. Entries in these buffers have time stamps (8-byte TOD clock values) that allow you to look at a dump and create a chronological list of the logrec events and WTO messages.

Reading the Logrec Recording Control Buffer

The logrec recording control buffer is a “wrap-table” similar to the system trace table. The entries are variable in size. The latest entries are the most significant especially if they have not yet been written to the logrec data set. Knowing the areas of the system that have encountered errors and the actions of their associated recovery routines, information obtained from the logrec data set and from the logrec recording control buffer helps provide an overall understanding of the environment you are about to investigate.

Note: The SDWA in the logrec buffer is a compressed SDWA in which the recordable extensions start directly after the used portion of the SDWAVRA. The SDWAURAL field contains the length of the SDWAVRA.

You can find the oldest entry in the buffer by locating the end of the unused or free area, obtained from RCBFREE+RCBFLNG. (If this sum brings you to a point beyond the end of the buffer, subtract RCBTLNG from the sum.) You can also read the buffer backwards by using the entry length at the end of each entry. The latest entry appears directly before the free or unused area of the buffer.

Reference

See *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)* for the format of the RCB.

Interpreting Software Records

There are two types of software records that are recorded in the logrec data set or the logrec log stream:

- **Software Record**

  The system generates these records, providing information from the system diagnostic work area (SDWA) that describes problems detected because of an abend or a program check. See “Detail Edit Report for a Software Record” on page 14-20 for more information.
- **Symptom Record**

  Either a user’s application program or the system can issue the SYMREC macro to request the creation of a symptom record. Generally, the symptom record describes problems not accompanied by an abend, but there are exceptions. See “Detail Edit Report for a Symptom Record” on page 14-26 for more information.

Using Report Information

Use the search argument you obtain from the detail edit reports for either a software record or a symptom record to search for a known problem. If you do not conduct the search yourself, contact the IBM Support Center. The result of the search will be one of the following:

- The PTF that corrects the problem.

  Apply the PTF that corrects the error.
- The APAR, and possibly the related APAR, that describes the problem. In some cases, a temporary fix (either ZAP or update) or a procedure might circumvent the problem.

  Apply the temporary fix if it is available; otherwise, follow the circumvention procedure.
- A description of why the problem might have occurred, which often describes a frequent misuse of a product that causes the error record. This type of problem is referred to as a user error.

  When an error occurs because of the misuse of a product other than MVS, use the procedures documented for that product to determine how best to debug the problem.

For any case other than the three listed above, including the case where the service link database does not contain a record matching the search criteria, contact the IBM Support Center to report the problem.

Detail Edit Report for a Software Record

The detail edit report for a software record shows the complete contents of an error record for an abnormal end, including the system diagnostic work area (SDWA). The report is produced by EREP and, through the VERBEXIT LOGDATA subcommand, under IPCS.

Use the detail edit report for a software record to determine the cause of an abend, and the recovery action that the system or application has either taken or not taken. This report enables you to locate where an error occurred, similar to the analysis of an SVC dump. Once you locate the error, you can develop a search argument to obtain a fix for the problem.

References

- See *EREP User's Guide* for information about producing a detail edit report for an SDWA-type record.
- See *z/OS MVS IPCS Commands* for information about the VERBEXIT LOGDATA subcommand.

Report Output

The example output is from one record. Following the two-part example is a list of the fields that are most important for diagnosis. Only the highlighted fields are described.

```
TYPE:  SOFTWARE RECORD      REPORT:  SOFTWARE EDIT REPORT           DAY .YEAR
       (SVC 13)                                        REPORT DATE: 105.92
SCP:   VS 2 REL 3                                       ERROR DATE: 093.92
                            MODEL:   9021                            HH:MM:SS.TH
                            SERIAL:  210142                    TIME: 20:14:10.99

JOBNAME:  SOFTREC
ERRORID:  SEQ=03283  CPU=0042  ASID=009C  TIME=20:14:10.7

SEARCH ARGUMENT ABSTRACT

  PIDS/5752SC1CM RIDS/IGC0101C#L RIDS/IEAVTSYM
AB/S0878 PRCS/00000010
  RIDS/SYMRCVR#R

  SYMPTOM             DESCRIPTION
  -------             -----------
  PIDS/5752SC1CM      PROGRAM ID: 5752SC1CM
  RIDS/IGC0101C#L     LOAD MODULE NAME: IGC0101C
  RIDS/IEAVTSYM       CSECT NAME: IEAVTSYM
  AB/S0878            SYSTEM ABEND CODE: 0878
  PRCS/00000010       ABEND REASON CODE: 00000010
  RIDS/SYMRCVR#R      RECOVERY ROUTINE
CSECT NAME: SYMRCVR

OTHER SERVICEABILITY INFORMATION

  DATE ASSEMBLED:          91085
  MODULE LEVEL:            UY62659
  SUBFUNCTION:             FAILURE IN EXT RTN

SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE

TIME OF ERROR INFORMATION

  PSW: 070C1000 811FAE8A   INSTRUCTION LENGTH: 02   INTERRUPT CODE: 000D
  FAILING INSTRUCTION TEXT: 00181610 0A0D18CE 18FB180C

  REGISTERS 0-7
  GR: 84000000 84878000 7F656E50 0000E512  007A9B10 007FDBD0 8127FFB8 00F84480
  AR: 012C3B10 00000000 00000000 00000000  00000000 00000000 00000000 00000000
  REGISTERS 8-15
  GR: 00000000 000008D8 00000000 7F656E20  00000040 7F656678 00FF1380 00000010
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000

  HOME ASID: 009C    PRIMARY ASID: 009C    SECONDARY ASID: 009C
  PKM: 0080          AX: 0000              EAX: 0000

  THE ERROR OCCURRED WHILE: A TYPE 1 SVC WAS IN CONTROL
                                             A LOCKED OR DISABLED ROUTINE WAS
 IN CONTROL
  NO LOCKS WERE HELD.
  NO SUPER BITS WERE SET.
```

```
STATUS FROM THE RB WHICH ESTABLISHED THE ESTAE EXIT

  PSW: 470C1000 841F03DC   INSTRUCTION LENGTH: 02   INTERRUPT CODE: 0023

RECOVERY ENVIRONMENT

  RECOVERY ROUTINE TYPE: ESTAE RECOVERY ROUTINE
  RECOVERY ROUTINE ENTRY POINT: 041F0534
  THE RB ASSOCIATED WITH THIS EXIT WAS NOT IN CONTROL AT THE TIME OF ERROR.
  USER REQUESTED NO I/O PROCESSING.

RECOVERY ROUTINE ACTION

  THE RECOVERY ROUTINE RETRIED TO ADDRESS 041F03E4.
  AN SVC DUMP WAS NOT REQUESTED.
  NO LOCKS WERE REQUESTED TO BE FREED.
  THE SDWA WAS REQUESTED TO BE FREED BEFORE RETRY.

  THE REGISTER VALUES TO BE USED FOR RETRY:
  REGISTERS 0-7
  GR: 00000000 0000004A 007B567D 00000001  00000004 00000010 00000010 7F656778
  AR: 012C3B10 00000000 00000000 00000000  00000000 00000000 00000000 00000000
  REGISTERS 8-15
  GR: 041F07C7 00000008 7F50C000 841EF7C8  7F656678 7F656678 7F50C2EC 00000000
  AR: 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000

HEXADECIMAL DUMP

  HEADER
  +000    40831820   00000000   0092093F   20141099    | C.......K.....R|
  +010    63210142   90210000                          |........        |

  JOBNAME
  +000    E2D6C6E3   D9C5C340                          |SOFTREC         |
```

```
SDWA BASE
+000    7F6567A0   04878000   FF04000D   00000000   |".. .G..........|
+010    FF850063   00000000   84000000   84878000   |.E......D...DG..|
                       .
                       .
                       .
+190    00FFA024                                    |.. .            |

VARIABLE RECORDING AREA (SDWAVRA)

+000    KEY: 10    LENGTH: 20
+002    841EF7C8   041F07C7   7F656678   007B567D   |D.7H...G"....#.'|
+012    7F656778   7F656678   00000000   40040000   |"..."....... ...|

+022    KEY: 53    LENGTH: 00

SDWA FIRST RECORDABLE EXTENSION (SDWARC1)
+000    E2C3F1C3   D4C6C1C9   D3E4D9C5   40C9D540   |SC1CMFAILURE IN |
                       .
                       .
                       .

SDWA SECOND RECORDABLE EXTENSION (SDWARC2)
                       .
                       .
                       .

SDWA THIRD RECORDABLE EXTENSION (SDWARC3)
                       .
                       .
                       .

ERRORID
+000    0CD30042   009C000B   1DBB                   |.L... ....     |
```

TYPE: SOFTWARE RECORD
Indicates that the detail edit report is for an SDWA-type record.

REPORT DATE
Indicates the date on which the EREP report was created.

ERROR DATE
Indicates the date on which the error occurred.

TIME
Indicates the time, as local, at which the error occurred.

JOBNAME
If the jobname is NONE-FRR, the error being recorded occurred in system or subsystem code covered by a functional recovery routine (FRR).

ERRORID
Allows you to coordinate diagnostic information from logrec, the console log (SYSLOG), and system dumps. The ERRORID is a concatenation of the following:

SEQ A unique number assigned to each error. The sequence number indicates the order of the errors, but the records might not be listed in order. It is important to scan all entries and examine the sequence numbers to understand which error occurred first.

You might find the same sequence number used in more than one entry when several recovery routines, as a result of percolation, get control and request recording for the same error; however, the error time stamp will be different.

CPU The internal identification number of the central processor that the failing process was running on at the time the error occurred. Use information from the system trace table about this CPU to learn more about the error.

ASID The address space identifier (ASID) of the current, or home, address space at the time the error occurred.

TIME Indicates the time of the error.

PIDS/... RIDS/... AB/... PRCS/...
Use this symptom string to do a structured search of any IBM database.

PROGRAM ID
The program ID (PID) indicates the product and the component where the error occurred. For IBM products, see the tables in *z/OS MVS Diagnosis: Reference* that list the products and components. For non-IBM products, see the appropriate vendor-supplied documentation.

LOAD MODULE NAME
Indicates the load module in control at the time of the error.

CSECT NAME
Supplied by the recovery routine that obtained control for the error. See the PSW for more information.

SYSTEM ABEND CODE
Indicates what system or user completion code was issued by the system, application, or component. See *z/OS MVS System Codes* for information about system abend codes. See the appropriate product documentation for user abend codes.

ABEND REASON CODE
Indicates the reason code, when available, associated with a system or user abend code.

RECOVERY ROUTINE CSECT NAME
Indicates the recovery routine that was given control to handle the error condition.

PSW
Indicates the program status word (PSW) at the time of the error.

If the software record is an SVC 13, the address in the second half of the PSW indicates the address of the module that detected the error. You need to find the caller of that module. The caller’s address will reside either in register 14, or, if register 14 points to module IEAVEEXP, use the STATUS section of the software record to determine the caller. In the STATUS section, the interrupt code will indicate the last SVC that was issued.

If the software record is a program interrupt, the address in the second half of the PSW usually points to the failing module.

FAILING INSTRUCTION TEXT
Contains 12 bytes of the instruction stream at the time of the error, including the actual instruction that caused the abend. Starting at the end of the sixth byte, subtract the instruction length to indicate the failing instruction. In the preceding example, the failing instruction is X'0A0D'.

THE ERROR OCCURRED WHILE . . .
Provides information about the system environment at the time of error, indicating what type of routine was in control, whether locks were held, and whether supervisor FRRs were set at the time of the error.

STATUS
The PSW and registers that follow come from the request block (RB) associated with the ESTAE recovery routine that obtained control for the error. Using the information indicated will enable you to determine the program that was running at the time of the error. This information included in the STATUS section does not appear when an FRR handles recovery.

RECOVERY ROUTINE ACTION
Describes the recovery action performed or requested to be performed by the recovery routine. In the preceding example, an SVC dump was not requested. There are times, however, when the recovery routine will request an SVC dump. If SVC DUMP SUCCESSFULLY STARTED appears in this section, the error identifier (ERROR ID) appears in the SVC dump and in message IEA911E as it appears in the logrec error record.

HEXADECIMAL DUMP
Provides an unformatted hexadecimal dump of the SDWA control block. Depending on an indicator in the SDWA, which is set by the recovery routine generating the record, the SDWA is displayed in hexadecimal; EBCDIC text; or key, length, and data format.

VARIABLE RECORDING AREA (SDWAVRA)
Provides component-specific information. Using the information in the PROGRAM ID field, determine the component. For IBM products, see *z/OS MVS Diagnosis: Reference* for diagnostic information related to system components.

The SDWAVRA can optionally be mapped in a key-length-data format. Recovery routines use the SDWAVRA to construct messages and provide data that often contains valuable debugging information. Some MVS recovery routines use the key-length-data format to provide standardized diagnostic information for software incidents. This formatted information allows you to screen duplicate errors.

Constants for the key field have been defined to describe data, such as: return and/or reason codes, parameter lists, registers, and control block information. For example, a key of X'10' indicates a recovery routine parameter area. The SDWAVRAM bit (in the fixed portion of the SDWA) indicates that the SDWAVRA has been mapped in the key-length-data format as described by the IHAVRA mapping macro.

Reference

See *z/OS MVS Data Areas, Vol 4 (RD-SRRA)* for the format of the SDWA, including a description of the keys.

SDWA RECORDABLE EXTENSIONS
In addition to the SDWA standard area and the SDWAVRA, the SDWA recordable extensions also contain valuable debugging information, as follows:

- *SDWARC1* (recording extension 1) contains additional component service data (such as the component ID, the component name, the address of the TCB representing the task that incurred the failure, the control registers, original completion code and reason code, linkage stack pointer, and translation exception access register number).

- *SDWARC2* (recording extension 2) contains additional I/O machine check data (such as the machine check interruption code).
- *SDWARC3* (recording extension 3) contains locking information (such as the locks to be freed, and the addresses of lockwords).

Note: The SDWA that is in the logrec buffer is a compressed SDWA in which the recordable extensions start directly after the used portion of the SDWAVRA. The SDWAURAL field contains the length of the SDWAVRA.

Detail Edit Report for a Symptom Record

The SYMREC macro updates a symptom record with system environment information and then logs the symptom record in the logrec data set or logrec log stream. The system or application, using the SYMREC macro, creates a symptom record. The ADSR mapping macro maps the symptom record, and the symptom record contains diagnostic information determined by the application.

As an application or a system component detects errors during processing, it stores diagnostic information into the symptom record and issues the SYMREC macro to log the record. The diagnostic information consists of a description of a programming failure and a description of the environment in which the failure occurred.

References

- See *z/OS MVS Programming: Assembler Services Reference IAR-XCT* for information about the SYMREC macro.
- See *z/OS MVS Data Areas, Vol 1 (ABEP-DALT)* for information about the ADSR data area.

Report Output: The following two-part example contains output from one record created by the system. Following the example is a list of the fields that are most important for diagnosis. Only the highlighted fields are discussed.

```
TYPE:  SYMPTOM RECORD       REPORT:  SOFTWARE EDIT REPORT             DAY.YEAR
                                                          REPORT DATE: 176.92
SCP:   VS 2 REL 3                                          ERROR DATE: 175.92
                            MODEL:   9021                              HH:MM:SS.TH
                            SERIAL:  031347                      TIME: 10:41:14.37

SEARCH ARGUMENT ABSTRACT:

    PIDS/5752SC1CM AB/S080A PRCS/00000010 RIDS/IEAVTRSR
    RIDS/IGC0101C#L FLDS/SR#ORIGIN VALU/CIEAVTRSR PCSS/FAILING

SYSTEM ENVIRONMENT:

    CPU MODEL:  9021              DATE:  175   92
    CPU SERIAL: 031347            TIME:  10:41:14.37
    SYSTEM:     CPUR              BCP:   MVS

    RELEASE LEVEL OF SERVICE ROUTINE:      JBB4422
    SYSTEM DATA AT ARCHITECTURE LEVEL:     10
    COMPONENT DATA AT ARCHITECTURE LEVEL:  10

    SYSTEM DATA:  00000000 00000000    |........|

COMPONENT INFORMATION:

    COMPONENT ID:            5752SC1CM
    COMPONENT RELEASE LEVEL:  D10
    DESCRIPTION OF FUNCTION:  RTM2 RECURSION ERROR RECORDING

PRIMARY SYMPTOM STRING:

    PIDS/5752SC1CM AB/S080A PRCS/00000010 RIDS/IEAVTRSR
    RIDS/IGC0101C#L FLDS/SR#ORIGIN VALU/CIEAVTRSR PCSS/FAILING
    PCSS/CSECT PCSS/UNKNOWN FLDS/RTM2SCTC FLDS/FROM#PRWA
    VALU/H00040000

    SYMPTOM            SYMPTOM DATA     EXPLANATION
    ---------------    -------------    -----------
    PIDS/5752SC1CM     5752SC1CM        COMPONENT IDENTIFIER
    AB/S080A           080A             ABEND CODE - SYSTEM
    PRCS/00000010      00000010         RETURN CODE
    RIDS/IEAVTRSR      IEAVTRSR         ROUTINE IDENTIFIER
    RIDS/IGC0101C#L    IGC0101C#L       ROUTINE IDENTIFIER
    FLDS/SR#ORIGIN     SR#ORIGIN        DATA FIELD NAME
    VALU/CIEAVTRSR     IEAVTRSR         ERROR RELATED CHARACTER VALUE
    PCSS/FAILING       FAILING          SOFTWARE STATEMENT
    PCSS/CSECT         CSECT            SOFTWARE STATEMENT
    PCSS/UNKNOWN       UNKNOWN          SOFTWARE STATEMENT
    FLDS/RTM2SCTC      RTM2SCTC         DATA FIELD NAME
    FLDS/FROM#PRWA     FROM#PRWA        DATA FIELD NAME
    VALU/H00040000     00040000         ERROR RELATED HEXADECIMAL VALUE
```

```
SECONDARY SYMPTOM STRING:

    FLDS/RTM2SCTR VALU/H00040000 FLDS/RTM2SCTX VALU/H00040000

    SYMPTOM            SYMPTOM DATA     EXPLANATION
    ---------------    -------------    -----------
    FLDS/RTM2SCTR      RTM2SCTR         DATA FIELD NAME
    VALU/H00040000     00040000         ERROR RELATED HEXADECIMAL VALUE
    FLDS/RTM2SCTX      RTM2SCTX         DATA FIELD NAME
    VALU/H00040000     00040000         ERROR RELATED HEXADECIMAL VALUE

FREE FORMAT COMPONENT INFORMATION:

    KEY = FF00     LENGTH = 00048 (0030)

    +000    C5D9D9D6   D940C4C5   E3C5C3E3   C5C440C2    |ERROR DETECTED B|
    +010    E840D9E3   D4F240D9   C5C3E4D9   E2C9E5C5    |Y RTM2 RECURSIVE|
                          .
                          .
                          .
HEX DUMP OF RECORD:

  HEADER

    +000    4C831800   00000000   0092175F   10411437    |<C.......K. ....|
    +010    A6031347   90210000                          |W.......        |

  SYMPTOM RECORD

    +000    E2D9F9F0   F2F1F0F3   F1F3F4F7   00000000    |SR9021031347....|
    +010    A5E2A254   A5ED4104   40404040   40404040    |VSS.V...        |
    +020    4040C3D7   E4D94040   4040F5F7   F5F2D1C2    |  CPUR    5752JB|
                           .
                           .
                           .
```

TYPE: SYMPTOM RECORD
Indicates that the detail edit report is for a symptom record.

SEARCH ARGUMENT ABSTRACT
Provides information you can use to create a search argument. If enough information exists in this field, you can search the IBM service link problem reporting database to determine if there is a PTF to correct the error.

The information that follows the search argument abstract in a symptom record depends on the options specified on the SYMREC macro either by a user program or by a system component. In the report output listed above, the system recorded a recursive error. The information contained in a symptom record is variable. To obtain an interpretation, contact the IBM Support Center for the product or for the component that built the record.

Customizing Symptom Record Location: You can control the location of logrec symptom records from non-authorized programs. Use the ASREXIT installation exit just before writing the logrec record to control:
- If a program can write symptom records
- The location of the symptom record: the logrec data set, job log, both, or neither

Reference

See *z/OS MVS Installation Exits* for information about ASREXIT.

Chapter 15. AMBLIST

The system's pasta maker. . . take a lump of dough, send it through AMBLIST, and you come out with beautiful spaghetti!

AMBLIST provides the following problem data:
- Formatted listing of an object module
- Map of the control sections (CSECTs) in a load module or program object
- List of modifications to the code in a CSECT
- Map of all modules in the link pack areas (LPA)
- Map of the contents of the DAT-on nucleus The map no longer represents the IPL version and message AMB129I will be issued.

These formatted listings can help you diagnose problems related to modules as they currently exist on your system. AMBLIST is a batch job that runs in problem state.

Major Topics

This chapter includes the following:
- "Obtaining AMBLIST Output"
- "Reading AMBLIST Output" on page 15-13

Long name support

AMBLIST will process external names (labels and references) up to 1024 bytes long. Names exceeding 16 bytes in length will be abbreviated in the formatted part of the listings and an abbreviation-to-long name equivalence table will be printed at the end of the listing. AMBLIST functions that provide long names support are: LISTLOAD, LISTIDR, and LISTOBJ (XSD and GOFF only).

Notes:
1. Any load module to be formatted and printed by AMBLIST must have the same format as those created by the linkage editor or by the program management binder.
2. Any program object to be formatted and printed by AMBLIST must have the same format as those created by the program management binder.
3. A program object format 2 or greater having the not-editable attribute cannot be listed by AMBLIST.

See "LISTLOAD OUTPUT=XREF Output (Comparison of Load Module and Program Object Version 1)" on page 15-44 for a comparison of the formatted output of a load module and a program object.

Obtaining AMBLIST Output

To obtain AMBLIST output, you must code JCL. Provide control statements as input to the job. These control statements dictate what type of information AMBLIST produces.

This section describes these topics:
- "Specifying the JCL Statements" on page 15-2
- "Controlling AMBLIST Processing" on page 15-2
- "Examples of Running AMBLIST" on page 15-6

Specifying the JCL Statements

Generally, the minimum partition or region for running AMBLIST is 64 kilobytes for all functions except LISTLPA, which requires 100 kilobytes. However,for large load modules, IBM recommends a minimum region size of 200 kilobytes. For program objects, IBM recommends a minimum region size of 12 megabytes.

AMBLIST requires the following JCL statements:

JOB
Initiates the job.

EXEC PGM=AMBLIST
Calls for the processing of AMBLIST.

SYSPRINT DD
Defines the message data set.

Anyname DD
Defines an input data set. This statement cannot define a concatenated data set.

SYSIN DD
Defines the data set (in the input stream) that contains AMBLIST control statements.

Controlling AMBLIST Processing

You control AMBLIST processing by supplying one or more control statements in the input stream. Code the control statement that applies to the data you want to obtain according to the following rules:

- Leave column 1 blank, unless you want to supply an optional symbolic name. A symbolic name is analogous to the label name in a program. The maximum length of a symbolic name is eight characters. A symbolic name must end with one or more blanks.
- If a complete control statement will not fit on a single line, end the first line with a comma or a non-blank character in column 72 and continue on the next line. Begin all continuation statements in columns 2 - 16. Do not split parameters between two lines. The only exceptions are the MEMBER parameters, which you can split at any internal comma.

LISTLOAD Control Statement

Use the LISTLOAD control statement to obtain a listing of load module or program objects. The listed data can help you verify why certain link-edit errors might have occurred.

```
LISTLOAD
 [OUTPUT={MODLIST|XREF|BOTH}]
  [,TITLE=('title',position)]
   [,DDN=ddname]
    [,MEMBER={member|(member1,membern...)}]
     [,RELOC=hhhhhhhh]
      [,ADATA={YES|NO}]
```

OUTPUT={MODLIST|XREF|BOTH}

OUTPUT=MODLIST requests a formatted listing of the text and control information of a load module or program object.

OUTPUT=XREF requests a module map and cross-reference listing for the load module or program object.

OUTPUT=BOTH requests both a formatted listing of the load module or program object and its map and cross-references.

If this parameter is omitted, OUTPUT=BOTH will be assumed.

TITLE=('title',position)

Specifies a title, from one to 40 characters long, to be printed below the heading line on each page of output. (The heading line identifies the page number and the type of listing being printed, and is not subject to user control.) The position subparameter specifies whether or not the title should be indented; if TITLE=('title',1) is specified, or if the position parameter is omitted, the title will be printed flush left, that is, starting in the first column. If you want the title indented from the margin, use the position parameter to specify the number of characters that should be left blank before the title. If you specify a position greater than 80, the indentation from the margin defaults to 1.

DDN=ddname

Identifies the DD statement that defines the data set containing the input object module. If the DDN= parameter is omitted, AMBLIST will assume SYSLIB as the default ddname.

MEMBER={member|(member1,membern...)}

Identifies the input load module or program object by member name or alias name. To specify more than one load module or program object, enclose the list of names in parentheses and separate the names with commas. If you omit the MEMBER= parameter, AMBLIST will print all modules in the data set.

Notes:

1. If you specify MEMBER=IEANUCxx, where xx is the suffix of the member used during the current IPL, AMBLIST will list the DAT-ON nucleus.
2. AMBLIST will accept member names up to 63 bytes in length. For aliases longer than 63 bytes, their primary member names must be entered instead.

RELOC=hhhhhhhh

Specifies a relocation or base address in hexadecimal of up to eight characters. When the relocation address is added to each relative map and cross-reference address, it gives the absolute central storage address for each item on the output listing. If you omit the RELOC parameter, no relocation is performed.

ADATA={YES|NO}

ADATA=YES on LISTLOAD OUTPUT=MODLIST or OUTPUT=BOTH requests a formatted listing of all ADATA classes, if exists, in the program object to be displayed in the traditional dump format, 32 bytes per line, with hexadecimal representation on the left and EBCDIC on the right, in addition to the output listing with the specified output parameter.

OUTPUT=NO on LISTLOAD OUTPUT=MODLIST or OUTPUT=BOTH requests a normal formatted listings with the specified output parameter, and ADATA suppressed.

If this parameter is omitted, ADATA=NO will be assumed.

LISTOBJ Control Statement

Use the LISTOBJ control statement to obtain listings of selected object modules. LISTOBJ supports traditional object modules as well as object modules in XOBJ or GOFF format.

```
LISTOBJ
 [TITLE=('title',position)]
  [,DDN=ddname]
   [,MEMBER={member|(member1,membern...)}]
```

TITLE=('title',position)
Specifies a title, from one to 40 characters long, to be printed below the heading line on each page of output. (The heading line identifies the page number and the type of listing being printed, and is not subject to user control.) The position parameter specifies whether or not the title should be indented; if TITLE=('title',1) is specified, or if the position parameter is omitted, the title will be printed flush left, that is, starting in the first column. If you want the title indented from the margin, use the position parameter to specify the number of characters that should be left blank before the title. If you specify a position greater than 80, the indentation from the margin defaults to 1.

DDN=ddname
Identifies the DD statement that defines the data set containing the input module. If the DDN parameter is omitted, AMBLIST will assume SYSLIB as the default ddname.

MEMBER={member|(member1[,membern]...)}
Identifies the input object module by member name or alias name. To specify more than one object module, enclose the list of names in parentheses and separate the names with commas.

Notes:

1. You must include the MEMBER parameter if the input object modules exist as members in a partitioned data set (PDS or PDSE). If you do not include the MEMBER parameter, AMBLIST will assume that the input data set is organized sequentially and that it contains a single, continuous object module.
2. AMBLIST will accept member names up to 63 bytes in length. For aliases longer than 63 bytes, their primary member names must be entered instead.

LISTIDR Control Statement

Use the LISTIDR control statement to obtain listings of selected CSECT identification records (IDR).

```
LISTIDR
 [OUTPUT={IDENT|ALL}]
  [,TITLE=('title',position)]
   [,DDN=ddname]
    [,MEMBER={member|(member1,membern...)}]
     [,MODLIB]
```

OUTPUT={IDENT|ALL}
Specifies whether AMBLIST should print all CSECT identification records or only those containing SPZAP data and user data. If you specify OUTPUT=ALL, all IDRs associated with the module will be printed. If you specify OUTPUT=IDENT, AMBLIST will print only those IDRs that contain SPZAP data or user-supplied data. If you omit this parameter, AMBLIST will assume a default of OUTPUT=ALL. Do not specify OUTPUT if you specify the MODLIB parameter.

TITLE=('title',position)
Specifies a title, from one to 40 characters long, to be printed below the heading line on each page of output. (The heading line identifies the page number and the type of listing being printed, and is not subject to user control.) The position parameter specifies whether or not the title should be indented; if TITLE=('title',1) is specified, or if the position parameter is omitted, the title is printed flush left, that is, starting in the first column. If you want the title indented from the margin, use the position parameter to specify the number of characters that should be left blank before the title. If a position greater than 80 is specified, the indentation from the margin defaults to 1.

DDN=ddname
Identifies the DD statement that defines the data set containing the input module. If you omit the DDN parameter, AMBLIST will assume SYSLIB as the default ddname.

MEMBER={member|(member1,membern...)}
Identifies the input load module or program object by member name or alias name. To specify more than one load module or program object, enclose the list of names in parentheses and separate the names with commas. If you omit the MEMBER parameter, AMBLIST will print all modules in the data set. Do not specify MEMBER if you specify the MODLIB parameter.

Note: AMBLIST will accept member names up to 63 bytes in length. For aliases longer than 63 bytes, their primary member names must be entered instead.

MODLIB
Prevents AMBLIST from printing the module summary. AMBLIST prints the IDRs that contain SPZAP data or user-supplied data. No page ejects occur between modules. When you specify MODLIB, the OUTPUT or MEMBER parameters are not valid parameters.

LISTLPA Control Statement

Use the LISTLPA control statement to obtain listings of selected modules in the fixed link pack area (LPA).

```
LISTLPA [FLPA][,MLPA][,PLPA]
```

LISTLPA
Lists the modules in the fixed link pack area, the modified link pack area, and the pageable link pack area (PLPA). This listing is a map that includes modules residing in the extended sections of each link pack area (LPA). If you do not specify any parameters on the LISTLPA control statement, then AMBLIST maps modules from all three LPAs.

Note: AMBLIST reflects only the system currently operating.

FLPA
Requests mapping of the modules in the fixed link pack area.

MLPA
Requests mapping of the modules in the modified link pack area.

PLPA
Requests mapping of the modules in the pageable link pack area.

Examples of Running AMBLIST

Using the control statements as input into the JCL for the job, you can invoke AMBLIST to provide output. The following examples of AMBLIST include the control statement needed to produce the output and sample JCL for each function.

List the Contents of an Object Module

You can use AMBLIST to format three types of object module:

1. OBJ (traditional object module)
2. XOBJ (extended object module, based on OBJ)
3. GOFF (Generalized Object File Format).

You can list the following information from an object module:

- the head record (HDR) - which may contain information about the character set and expected operating environment (GOFF only)
- external symbol dictionary entries (ESD or XSD)
- relocation dictionary entries (RLD)
- the text of the program - the instructions and data, as output by the language translator (TXT)
- translator identification record (IDRL) - which contains the compiler ID and compile date
- ADATA records (GOFF only)
- LEN records (GOFF only)
- and the END record.

To list object module contents, invoke AMBLIST with the LISTOBJ control statement.

For sample outputs, see “LISTOBJ Outputs” on page 15-18.

Example: Listing an Object Module

In this example, AMBLIST is used to format and list an object module included in the input stream.

```
//LSTOBJDK     JOB      MSGLEVEL=(1,1)
//             EXEC     PGM=AMBLIST,REGION=64K
//SYSPRINT     DD       SYSOUT=A
//OBJMOD       DD       *
    object module
/*
//SYSIN        DD       *
    LISTOBJ         DDN=OBJMOD,
        TITLE=('OBJECT MODULE LISTING FOR MYJOB',25)
/*
```

OBJMOD DD Statement
Defines the input data set, which follows immediately. In this case, the input data set is an object module.

SYSIN DD Statement
Defines the data set containing AMBLIST control statements, which follows immediately.

LISTOBJ Control Statement
Instructs AMBLIST to format the data set defined by the OBJMOD DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

Example: Listing Several Object Modules

In this example, AMBLIST is used to list all object modules contained in the data set named OBJMOD, and three specific object modules from another data set called OBJMODS.

Note: If you are using AMBLIST to list program objects, IBM recommends that you specify REGION=12M or higher.

```
//OBJLIST      JOB    MSGLEVEL=(1,1)
//LISTSTEP     EXEC   PGM=AMBLIST,REGION=64K
//SYSPRINT     DD     SYSOUT=A
//OBJLIB       DD     DSN=OBJMODS,DISP=SHR
//OBJSDS       DD     DSN=OBJMOD,DISP=SHR
//SYSIN        DD     *
    LISTOBJ        DDN=OBJSDS,
        TITLE=('OBJECT MODULE LISTING OF OBJSDS',20)
    LISTOBJ        DDN=OBJLIB,MEMBER=(OBJ1,OBJ2,OBJ3),
        TITLE=('OBJECT MODULE LISTING OF OBJ1 OBJ2 OBJ3',20)
/*
```

OBJLIB and OBJSDS DD Statements
Define input data sets that contain object modules.

SYSIN DD Statement
Defines the data set in the input stream containing AMBLIST control statements.

LISTOBJ Control Statement #1
Instructs AMBLIST to format the data set defined by the OBJSDS DD statement, treating it as a single member. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTOBJ Control Statement #2
Instructs AMBLIST to format three members of the partitioned data set (PDS or PDSE) defined by the OBJLIB DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

Map the CSECTs in a Load Module or Program Object

You can list the organization of CSECTs within the load module or program object, the overlay structure (if any), and the cross-references for each CSECT.

To map CSECTs, invoke AMBLIST with the LISTLOAD control statement.

For sample output, see “LISTLOAD OUTPUT=MODLIST Output” on page 15-28, “Alphabetical Cross-Reference” on page 15-42, and “LISTLOAD OUTPUT=XREF Output (Comparison of Load Module and Program Object Version 1)” on page 15-44.

Example: Listing Several Load Modules or Program Objects

In this example, AMBLIST is used to produce formatted listings of several load modules or program objects.

Note: If you are using AMBLIST to format program objects, IBM recommends that you specify REGION=2M or higher.

```
//LOADLIST   JOB         MSGLEVEL=(1,1)
//LISTSTEP   EXEC        PGM=AMBLIST,REGION=64K
//SYSPRINT   DD          SYSOUT=A
//SYSLIB     DD          DSNAME=SYS1.LINKLIB,DISP=SHR
//LOADLIB    DD          DSNAME=LOADMOD,DISP=SHR
//SYSIN      DD          *
    LISTLOAD OUTPUT=MODLIST,DDN=LOADLIB,
        MEMBER=TESTMOD,
        TITLE=('LOAD MODULE LISTING OF TESTMOD',20)
    LISTLOAD OUTPUT=XREF,DDN=LOADLIB,
        MEMBER=(MOD1,MOD2,MOD3),
        TITLE=('XREF LISTINGS OF MOD1 MOD2 AND MOD3',20)
    LISTLOAD TITLE=('XREF&LD MOD LSTNG-ALL MOD IN LINKLIB',20)
/*
```

SYSLIB DD Statement
Defines an input data set, SYS1.LINKLIB, that contains load modules or program objects to be formatted.

LOADLIB DD Statement
Defines a second input data set.

SYSIN DD Statement
Defines the data set (in the input stream) containing the AMBLIST control statements.

LISTLOAD Control Statement #1
Instructs AMBLIST to format the control and text records including the external symbol dictionary and relocation dictionary records of the load module or program object TESTMOD in the data set defined by the LOADLIB DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTLOAD Control Statement #2
Instructs AMBLIST to produce a module map and cross-reference listing of the load modules or program objects MOD1, MOD2, and MOD3 in the data set defined by the LOADLIB DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTLOAD Control Statement #3
Instructs AMBLIST to produce a formatted listing of the load module or program object and its map and cross-reference listing. Because no DDN= parameter is included, the input data set is assumed to be the one defined by the SYSLIB DD statement. Because no MEMBER parameter is specified, all load modules in the data set will be processed. This control statement also specifies a title for each page of output, to be indented 20 characters from the left margin.

Example: Listing Several Load Modules or Program Objects

This example shows how to use AMBLIST to verify three modules. Assume that an unsuccessful attempt has been made to link-edit an object module with two load modules or program objects to produce one large load module or program object.

```
>
   //LSTLDOBJ      JOB     MSGLEVEL=(1,1)
   //              EXEC    PGM=AMBLIST,REGION=64K
   //SYSPRINT      DD      SYSOUT=A
   //OBJMOD        DD      DSN=MYMOD,DISP=SHR
   //LOADMOD1      DD      DSN=YOURMOD,DISP=SHR
   //LOADMOD2      DD      DSN=HISMOD,DISP=SHR
   //SYSIN         DD      *
       LISTOBJ         DDN=OBJMOD,
           TITLE=('OBJECT LISTING FOR MYMOD',20)
       LISTLOAD        DDN=LOADMOD1,OUTPUT=BOTH,
           TITLE=('LISTING FOR YOURMOD',25)
       LISTIDR         DDN=LOADMOD1,OUTPUT=ALL,
           TITLE=('IDRS FOR YOURMOD',25)
       LISTLOAD        DDN=LOADMOD2,OUTPUT=BOTH,
           TITLE=('LISTING FOR HSMOD',25)
       LISTIDR         DDN=LOADMOD2,OUTPUT=ALL,
           TITLE=('IDRS FOR HISMOD',25)
```

OBJMOD DD Statement
Defines an input load module or program object data set.

LOADMOD1 and LOADMOD2 DD Statements
Define input load module or program object data sets.

SYSIN DD Statement
Defines the data set containing AMBLIST control statements.

LISTOBJ Control Statement
Instructs AMBLIST to format the data set defined by the OBJMOD DD statement. It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTLOAD Control Statement #1
Instructs AMBLIST to format all records associated with the data set defined by the LOADMOD1 DD statement. It also specifies a title for each page of output, to be indented 25 characters from the left margin.

LISTIDR Control Statement #1
Instructs AMBLIST to list all CSECT identification records associated with the data set defined by the LOADMOD1 DD statement. It also specifies a title for each page of output, to be indented 25 characters from the left margin.

LISTLOAD Control Statement #2
Instructs AMBLIST to format all records associated with the data set defined by the LOADMOD2 DD statement. It also specifies a title for each page of output, to be indented 25 characters from the left margin.

LISTIDR Control Statement #2
Instructs AMBLIST to list all CSECT identification records associated with the data set defined by the LOADMOD2 DD statement. It also specifies a title for each page of output to be indented 25 characters from the left margin.

Trace Modifications to the Executable Code in a CSECT

You can list the information in a load module or program object's CSECT identification records (IDRs). An IDR provides the following information:

- The version and modification level of the language translator and the date that each CSECT was translated. (Translation data is available only for CSECTs that were produced by a translator that supports IDR generation.)
- The version and modification level of the linkage editor or binder that built the load module or program object and gives the date the load module or program object was created.
- Modifications to the load module or program object, by date, that might have been done using SPZAP.

An IDR might also contain optional user-supplied data.

To trace modifications, invoke AMBLIST with the LISTIDR control statement.

For sample output, see "LISTIDR Output" on page 15-47.

Example: Listing IDR Information for Several Load Modules

In this example, AMBLIST is used to list the CSECT identification records in several load modules or program objects.

```
//IDRLIST     JOB     MSGLEVEL=(1,1)
//LISTSTEP    EXEC    PGM=AMBLIST,REGION=64K
//SYSPRINT    DD      SYSOUT=A
//SYSLIB      DD      DSN=SYS1.LINKLIB,DISP=SHR
//LOADLIB     DD      DSN=LOADMODS,DISP=SHR
//SYSIN       DD      *
    LISTIDR   TITLE=('IDR LISTINGS OF ALL MODS IN LINKLIB',20)
    LISTIDR   OUTPUT=IDENT,DDN=LOADLIB,MEMBER=TESTMOD
              TITLE=('LISTING OF MODIFICATIONS TO TESTMOD',20)
    LISTIDR   OUTPUT=ALL,DDN=LOADLIB,MEMBER=(MOD1,MOD2,MOD3),
              TITLE=('IDR LISTINGS OF MOD1 MOD2 MOD3',20)
    LISTIDR   DDN=LOADLIB,MODLIB
/*
```

SYSLIB DD Statement
Defines an input data set, SYS1.LINKLIB, that contains load modules or program objects to be processed.

LOADLIB DD Statement
Defines a second input data set.

SYSIN DD Statement
Defines the data set (in the input stream) containing the AMBLIST control statements.

LISTIDR Control Statement #1
Instructs AMBLIST to list all CSECT identification records for all modules in SYS1.LINKLIB (this is the default data set since no DDN parameter was included). It also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTIDR Control Statement #2
Instructs AMBLIST to list CSECT identification records that contain SPZAP or user-supplied data for the load module or program object named TESTMOD. TESTMOD is a member of the data set defined by the LOADLIB DD statement. This control statement also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTIDR Control Statement #3
Instructs AMBLIST to list all CSECT identification records for of the load modules or program objects MOD1, MOD2, and MOD3. These are members in the data set defined by the LOADLIB DD statement. This control statement also specifies a title for each page of output, to be indented 20 characters from the left margin.

LISTIDR Control Statement #4
Instructs AMBLIST to list CSECT identification records that contain SPZAP or user-supplied data for the LOADLIB data set. The module summary print out is suppressed.

List the Modules in the Link Pack Area and the Contents of the DAT-On Nucleus

You can list all modules in the fixed link pack area, the modified link pack area, and the pageable link pack area.

To map link pack area modules, invoke AMBLIST with the LISTLPA control statement.

For sample output, see “LISTLPA Output” on page 15-50.

You can also produce a map and cross-reference listing of a nucleus.

To map the contents of the DAT-on nucleus, invoke AMBLIST with the LISTLOAD MEMBER=IEANUCxx control statement.

For sample output, see “LISTLOAD Output: DAT-ON Nucleus” on page 15-51.

Example: Listing a System Nucleus and the Link Pack Area

This example shows how to use the LISTLOAD and LISTLPA control statements to list a system nucleus and map the fixed link pack area, the modified link pack area, and the pageable link pack area. Note that in this example the data set containing the nucleus is named SYS1.NUCLEUS, and the nucleus occupies the member named IEANUC01. The map no longer represents the IPL version of the nucleus and message AMB129I will be issued. Use IPCS to format the NUCMAP. For information on using IPCS see the *z/OS MVS IPCS User’s Guide* and the *z/OS MVS IPCS Commands*.

```
//LISTNUC      JOB     MSGLEVEL=(1,1)
//STEP         EXEC    PGM=AMBLIST,REGION=100K
//SYSPRINT     DD      SYSOUT=A
//SYSLIB       DD      DSN=SYS1.NUCLEUS,DISP=SHR,UNIT=3330,
//    VOL=SER=nnnnn
//SYSIN        DD      *
    LISTLOAD           DDN=SYSLIB,MEMBER=IEANUC01,
       TITLE=('LISTING FOR NUCLEUS IEANUC01',25)
    LISTLPA
/*
```

SYSLIB DD Statement
Defines the input data set, which in this case contains the nucleus.

SYSIN DD Statement
Defines the data set containing AMBLIST control statements, which follows immediately.

LISTLOAD Control Statement
Instructs AMBLIST to format the control and text records including the external symbol dictionary and relocation dictionary records of the load module IEANUC01 in the data set defined by the SYSLIB DD statement. It also specifies a title for each page of output, to be indented 25 characters from the left margin.

LISTLPA Control Statement
Instructs AMBLIST to map the fixed link pack area (FLPA), the modified link pack area (MLPA), and the pageable link pack area (PLPA).

Reading AMBLIST Output

AMBLIST produces a separate listing for each control statement that you specify.

- The first page of each listing always shows the control statement as it was entered.

- The second page of the listing is a module summary, unless you requested LISTOBJ, LISTLPA, or MODLIB with LISTIDR; in that case, no module summary will be produced, and the second page of the listing will be the beginning of the formatted output.

  The module summary gives the member name (with aliases), the entry point and its addressing mode, alias entry points and their addressing modes, the attributes assigned to the module by the linkage editor or program management binder, the system status index information (SSI), the APF code, an residence mode for the module being formatted. For program objects, the PMAR and PMARL are displayed in hexadecimal for diagnostic information. Figure 15-1 and Figure 15-2 on page 15-15 show samples of module summary processed by the Linkage Editor and the Binder.
- The third page of the listing (or, for LISTOBJ, LISTLPA, or MODLIB with LISTIDR the second page) is the beginning of the formatted output itself.

Module Summary

```
 LISTLOAD DDN=DD1,MEMBER=TESTPR

A                       ***** M O D U L E   S U M M A R Y *****
        MEMBER NAME:  TESTPR                                                          MAIN ENTRY POINT:    00000000
        LIBRARY:      DD1                                                            AMODE OF MAIN ENTRY POINT: ANY
        NO ALIASES **
------------------------------------------------------------------------------------------------------------------------

B       ****   LINKAGE EDITOR ATTRIBUTES OF MODULE   *
                **   BIT  STATUS        BIT  STATUS        BIT  STATUS        BIT  STATUS   **
                      0  NOT-RENT         1  NOT-REUS        2  NOT-OVLY        3  NOT-TEST
                      4  NOT-OL           5  BLOCK           6  EXEC            7  MULTI-RCD
                      8  NOT-DC           9  ZERO-ORG       10 EP-ZERO         11 RLD
                      12 EDIT            13 NO-SYMS         14 F-LEVEL         15 NOT-REFR
------------------------------------------------------------------------------------------------------------------------

C     MODULE SSI:  NONE
                              APFCODE:     00000000
                              RMODE:       24

D      *****LOAD MODULE PROCESSED BY VS LINKAGE EDITOR
```

Figure 15-1. Sample Module Summary for a Load Module Processed by the Linkage Editor

```
 LISTLOAD DDN=DD1,MEMBER=B#Z49$EA

A      *****  M O D U L E   S U M M A R Y  *****
   MEMBER NAME:  B#Z49$EA                                                          MAIN ENTRY POINT:    00000000
   LIBRARY:      MYLIB                                                             AMODE OF MAIN ENTRY POINT: 31
       ** ALIASES **      ENTRY POINT    AMODE
         A1                00000000       31
         A2                00000000       31
      ** A3                00000000       31
         A4                00000000       31
         THIS#IS#A-BF6190  00000000       31   ALT PRIMARY
-------------------------------------------------------------------------------------------------------------------------
B         **** ATTRIBUTES OF MODULE ***
               **   BIT  STATUS         BIT  STATUS          BIT  STATUS          BIT  STATUS   **
                     0  NOT-RENT         1  NOT-REUS          2  NOT-OVLY          3  NOT-TEST
                     4  NOT-OL           5  BLOCK             6  EXEC              7  MULTI-RCD
                     8  DC               9  ZERO-ORG          10 RESERVED          11 RLD
                     12 EDIT             13 NO-SYMS           14 RESERVED          15 NOT-REFR
                     16 RESERVED         17 <16M              18 NOT-PL            19 NO-SSI
                     20 NOT-APF          21 PGM OBJ           22 RESERVED          23 RESERVED
                     24 ALTP             25 RESERVED          26 RESERVED          27 RMODE24
                     28 RESERVED         29 RESERVED          30 RESERVED          31 RESERVED
                     32 MIGRATE          33 NO-PRIME          34 NO-PACK           35 RESERVED
                     36 RESERVED         37 RESERVED          38 RESERVED          39 RESERVED
-------------------------------------------------------------------------------------------------------------------------

C          MODULE SSI:  NONE
                                APFCODE:     00000000
                                RMODE:       24
                      PO FORM:     2
                                XPLINK:      NO


D           *****PROGRAM OBJECT PROCESSED BY BINDER


E
***THE FOLLOWING ARE THE UNFORMATTED PDSE DIRECTORY ENTRY SECTIONS (PMAR AND PMARL)
PMAR  001E0206 02400400 00000000 25800000   00000000 00000000 00000000 0000
PMARL 00520002 FF000000 00060000 25800000   10000000 08680000 48300000 008C0000
      01280000 00240000 01040000 00060000   01B40001 00000000 25800000 00000000
      00001995 153F0131 714FD7D4 E2E3D7F4   F0F1
```

Figure 15-2. Sample Module Summary for a Program Object Processed by the Binder

The following describes Figure 15-1 on page 15-14 and Figure 15-2.

A Entry Names

For the member, the library (ddname) and member name are displayed, along with the primary entry point offset and AMODE. The MEMBER NAME field contains the primary name.

For each alias or alternate entry point, AMBLIST shows the alias name, entry point offset and AMODE. If no aliases are present, AMBLIST prints NO ALIASES. If the input name is an alias, then its name is printed in the alias section preceded by two asterisks.

The constants ALT PRIMARY will be added to the right of the amode of the alias name which was a long primary name in which the binder had converted to an alias.

B Attributes of the Module

The attributes of the module are represented by bits. Each bit is set either ON or OFF. In the listing, AMBLIST interprets the bit settings and shows a descriptive value in the STATUS column. For example, in Figure 15-1 on page 15-14 and Figure 15-2, bit 0 is interpreted as NOT-RENT. This means the module is not reentrant. For a description of all the STATUS values, see Table 15-1 on page 15-16.

C Other Attributes

The remaining module attributes are displayed following the table. This includes the system status index (SSI) field, the APF (authorized program facility) code, the RMODE (residence mode) for the entire module. PO format and XPLINK are applicable only for program objects. PO format is the version of the program object. XPLINK indicates whether any routines use XPLINK linkage conventions. If attribute bit 19 is the OFF state, NONE will be displayed in place of SSI. (SSI is usually set via the SETSSI control statement in the binder or SPZAP programs.)

D Linking Program

The last line in the module summary identifies the linking program (VS linkage editor or binder) that created the module. For example:

```
*****PROGRAM OBJECT PROCESSED BY BINDER
```

This is applicable to a program object.

```
****LOAD MODULE PROCESSED EITHER BY VS LINKAGE EDITOR OR BINDER
```

The load module is either created by the linkage editor and processed by the binder, or created by the binder and processed by the linkage editor.

E PMAR and PMARL

For program objects, the PMAR and PMARL are displayed in hexadecimal for diagnostic purposes, preceded by:

```
*** THE FOLLOWING ARE THE UNFORMATTED PDSE DIRECTORY ENTRY SECTIONS
(PMAR AND PMARL)
```

Table 15-1. Program Object and Load Module Attributes. The first column shows the bit position. Columns 2 and 3 show the displayed constant and its meaning for the OFF condition. Columns 4 and 5 show the displayed constant and meaning for the ON condition.

Bit Position	OFF Value	Meaning	ON Value	Meaning
00	NOT-RENT	Module is not reentrant.	RENT	Module is reentrant.
01	NOT-REUS	Module is not reusable.	REUS	Module is reusable.
02	NOT-OVLY	Module is not in overlay format.	OVLY	Module is in overlay format.
03	NOT-TEST	Test option was not specified during binding.	TEST	Test option was specified during binding.
04	NOT-OL	Program can be invoked via all CSV macros.	ONLY-LOAD	Program can be loaded only via LOAD macro.
05	BLOCK	Module consists of a single, contiguous block of text. This bit is always off for program objects.	SCTR	Module can be scatter loaded (MVS nucleus only).
06	NON-EXEC	Module is marked not executable.	EXEC	Module is marked executable.
07	MULTI-RCD	Module contains multiple text records. This bit is always off for program objects.	1-TXT	Module contains no RLD items and only one block of text.

Table 15-1. Program Object and Load Module Attributes (continued). The first column shows the bit position. Columns 2 and 3 show the displayed constant and its meaning for the OFF condition. Columns 4 and 5 show the displayed constant and meaning for the ON condition.

Bit Position	OFF Value	Meaning	ON Value	Meaning
08	DC	Module is processable by all levels of linkage editor.	NOT-DC	Module is processable only by F-level linkage editor and above. This bit is always on for program objects.
09	NOT-ZERO	Origin of first text block greater than zero.	ZERO-ORG	Origin of first text block is zero. This bit is always on for program objects.
10	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
11	RLD	Module contains RLD items.	NO-RLD	Module contains no RLD items.
12	EDIT	Module can be reprocessed by binder.	NOT-EDIT	Module cannot be reprocessed by binder.
13	NO-SYMS	Module contains no SYM records.	SYMS	Module contains SYM records.
14	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
15	NOT-REFR	Module is not refreshable.	REFR	Module is refreshable.
16	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
Note: The following bits are shown only for program objects.				
17	<16M	Module text size is less than 16 megabytes.	>16M	Module text size is greater than or equal to 16 megabytes.
18	NOT-PL	Page alignment is not required for loaded text.	P-ALIGN	Page alignment is required for loaded text.
19	NO-SSI	System status index is not present.	SSI	System status index is present.
20	NOT-APF	Module is not APF-authorized.	APF	Module is APF-authorized.
21	NOT-PO	This is a load module.	PGM OBJ	This is a program object. Always on for program object.
22	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
23	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
24	ALTP	Alternate primary name.	NOT-ALTP	Not an alternate primary name.
25	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.

Table 15-1. Program Object and Load Module Attributes (continued). The first column shows the bit position. Columns 2 and 3 show the displayed constant and its meaning for the OFF condition. Columns 4 and 5 show the displayed constant and meaning for the ON condition.

Bit Position	OFF Value	Meaning	ON Value	Meaning
26	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
27	RMODE24	Module must be loaded below 16 megabytes.	RMODEANY	Module can be loaded anywhere below 2 gigabytes.
28	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
29	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
30	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
31	RESERVED	Reserved for IBM use.	RESERVED	Reserved for IBM use.
32	NON-MIGR	Program object cannot be converted directly to PDS load module format.	MIGRATE	Program object can be converted to PDS load module format.
33	PRIME	FETCHOPT PRIME option.	NO-PRIME	FETCHOPT NOPRIME option.
34	PACK	FETCHOPT PACK option.	NO-PACK	FETCHOPT NOPACK option.

LISTOBJ Outputs

```
OBJECT MODULE LISTING                                                                                              PAGE 0001

ESD RECORD:                                                                                                        00000001
  ESDID   TYPE     NAME      ADDR   R/R/A  ID/LTH
   0001   SD(00)   RDONLYB1  000000   03   00002C
   0002   ER(02)   RDONLYB2  000000   40   404040
   0003   ER(02)   RDONLYE1  000000   40   404040

ESD RECORD:                                                                                                        00000002
  ESDID   TYPE     NAME      ADDR   R/R/A  ID/LTH
   0004   ER(02)   RDONLYE2  000000   40   404040
   0005   ER(02)   RDWRTE01  000000   40   404040
   0006   ER(02)   RDWRTE02  000000   40   404040

ESD RECORD:                                                                                                        00000002
  ESDID   TYPE     NAME      ADDR   R/R/A  ID/LTH
          LD(01)   LYB1      00000C   40   000001

TXT:                                                                                                               00000004
  ADDR=000000  ESDID=0001 TEXT:  90CED000  05C098CE  D00007FE  90CED000  05C098CE  D00007FE  00000000  00000000  00000000  00000000
                                 00000000

RLD RECORD:   R PTR   P PTR    FLAGS   ADDR    R PTR   P PTR   FLAGS   ADDR    R PTR   P PTR   FLAGS   ADDR       00000000
              0002    0001      1C     000018  0003    0001     1C     00001C  0004    0002     1C     000020
              0005    0001      1C     000024  0006    0001     1C     000028
END RECORD:                                                          15741SC103 020180171
```

Figure 15-3. Sample Output for LISTOBJ with an Object Module

The record formats for OBJ and XOBJ records are identical except that XOBJ modules contain XSD records rather than ESD records. Except for XSD records, AMBLIST formats the records in the object module one record at a time. XSD records support names up to 1024 characters. These names may be continued onto multiple records, but such a continued record will appear as a single XSD record in the AMBLIST output. If the name is longer than 16 characters, a 16–character abbreviated name is printed with the XSD record. An abbreviation table which maps abbreviated names to be true names is printed at the end of the listing.

Reference

See the description of ESD data items in *z/OS DFSMS Program Management* for a description of the format of OBJ and XOBJ record formats.

```
OBJECT MODULE LISTING                             MEMBER= CALLEDA                                                             PAGE 0001
             LIST OF CALLEDA
XSD RECORD:                                                                                                                    00000001
 ESDID    TYPE      NAME             ADDR    R/R/A   ID/LTH
  0001   SD(00)   CAN_BE_ABBRV_16B  000000    06     0000BC
TXT:                                                                                                                           00000002
 ADDR=000000 ESDID= 0001 TEXT: 90ECD00C 0DC050D0 C07241E0 C06E50E0 D00818DE 1B115010 C0660700 4510C048 80000070 00000003
                               02250000 C3C1D3D3 C5C4C140 C1C2D6E4
TXT:                                                                                                                           00000004
 ADDR=000038 ESDID= 0001 TEXT: E340E3D6 40D9C5E3 E4D9D540 E3D640C3 C1D3D3C5 D9
TXT:                                                                                                                           00000005
 ADDR=00004E ESDID= 0001 TEXT: 0A234100 00014110 C0660A01 1BFF58D0 D00458E0 D00C980C D0140B0E 00000000 0000E7E7
RLD RECORD:    R PTR   P PTR   FLAGS   ADDR   R PTR   P PTR   FLAGS   ADDR   R PTR   P PTR   FLAGS   ADDR       00000006
               0001    0001     0D    000020  0001    0001     0C    000024
END RECORD:                                                        1566896201 020191248                                        00000007
```

Figure 15-4. Sample Output for LISTOBJ with XSD Record

```
                      ***** G E N E R A L I Z E D   O B J E C T   F I L E   F O R M A T *****               PAGE       1

OBJECT MODULE LISTING

RECORD TYPE: HDR      SEQUENCE:        1
           --- CHARACTER SET --    LANGUAGE
           ID         NAME          PRODUCT

>         00123                    FORTRAN/90

RECORD TYPE: ESD      SEQUENCE:        2
        ESD   OWNER/    ITEM      ITEM    NAME   ---------------- ATTRIBUTES -----------------
 ESDID  TYPE  PARENT   OFFSET    LENGTH   SP/S   BA  AMD  RMD  REUS  AL  TXT  ORD  STR  BINDER            SIGNATURE

>000001  SD     N/A      N/A       N/A    N/A    N/A N/A  N/A   N/S  N/A N/A  N/A  N/A  N/A                 N/A
              NAME(CSECT)

>000002  ED   000001        0        1C  01-N/A C   N/A  N/S   N/A  03  B-U  N/A  N/A  L,A,C              N/A
              NAME(B_TEXT)

>000004  ED   000001        0         0  01-N/A C   N/A  N/S   N/A  00  F-U  N/A  N/A  C                   N/A
              NAME(B_IDRL)

>000003  LD   000002        0     N/A    01-N/S N/A ANY  N/A   N/A  N/A N-U  N/A   S   N/A              00000000
              NAME(CSECT)

RECORD TYPE: TEXT     SEQUENCE:        6
 -- RESIDENT --     TRUE      TEXT    ENCODED
 ESDID   OFFSET    LENGTH   ENCODING  LENGTH    ------------------------------ T E X T  ------------------------------

>000002 00000000  00000000    0000   0000000C   58C07010 58C07014 41C07018

>000002 00000010  00000000    0000   0000000C   00000001 00000004 0000001F

RECORD TYPE: RLD      SEQUENCE:        8
 R-PTR  P-PTR  OFFSET  TYPE LEN ATTRIB   R-PTR  P-PTR  OFFSET  TYPE LEN ATTRIB   R-PTR  P-PTR  OFFSET  TYPE LEN ATTRIB
>000002 000002 000010   00+ 004          000002 000002 000014   00+ 004          000002 000002 000018   00+ 004

RECORD TYPE: IDRL     SEQUENCE:        9
 ESDID             |---- IDR  DATA ----|    |---- IDR  DATA ----|    |---- IDR  DATA ----|    |---- IDR  DATA ----|

>000004            |569623400.010295104|

RECORD TYPE: END      SEQUENCE:       10
 RECORD       --ENTRY POINT--
 COUNT        ESDID    OFFSET

>000000        N/S       N/S
```

Figure 15-5. Sample Output for LISTOBJ with GOFF Records

Description of LISTOBJ Output for GOFF

The GOFF object listing is similar in function and content to the LISTOBJ format for traditional object modules. The output is formatted one logical record at a time. A logical record represents the concatenation of the first physical record (which contains the record type) and all continuation records. If a name in a record is longer than 16 characters, a 16-character abbreviated name is printed. The true name can be found from the abbreviated name to long name table, which is printed at the end of the listing. The start of a logical record is highlighted by a dingbat (“>”) in the first position.

A record group consists of one or more records of the same type and is preceded by a two- or three-line record header. The first line of each record header consists of the record type and the sequence number of the first record in the group. Following a page break, the record group header will be repeated, even though the record type may not have changed.

Although the GOFF format currently defines only six record types, the TXT record type is subdivided into three different text types:

- TEXT, containing the instructions and data of the program
- IDRL, containing IDR information from the compiler or assembler
- ADATA, containing additional data associated with the object module

Altogether there are eight different display formats.

Report Description

The keyed sections of this description correspond to the equivalent keys highlighting the page header and the eight record formats in Figure 15-6 on page 15-22. Note that some of the flags and lengths in the GOFF format are of a structural nature and do not represent the data content of the module. To save space, those elements have been omitted from the listing. For the same reason, unsupported data elements are not shown. A list of omitted elements is provided for each record type and the reason for omission is coded in parens following the field name. Code values are S (structural or syntactic data) and U (unsupported element). PTV for all record types is not formatted.

```
                      ***** G E N E R A L I Z E D  O B J E C T  F I L E  F O R M A T *****             PAGE        1
                                      MY PROGRAM IN GOFF FORMAT     1
 OBJECT MODULE LISTING
 RECORD TYPE: HDR      SEQUENCE:        1   2
            --- CHARACTER SET --    LANGUAGE
            ID         NAME          PRODUCT

 >        00000

0RECORD TYPE: ESD      SEQUENCE:        2   3
        ESD   OWNER/    ITEM      ITEM     NAME   --------------------------- ATTRIBUTES ---------------------------
  ESDID  TYPE  PARENT   OFFSET   LEN/ADA   SP/S   BA  AMD  RMD  REUS  AL  TXT  ORD  STR  BINDER_FLAGS  LNK SIGNATURE

 >000001  SD   000000     N/A       N/A     N/A    N/A N/S  N/A  RENT  N/A N/A  N/A  N/A  N/A           N/A   N/A
              NAME( )

 >000002  ED   000001         0        28  01-N/A C    N/A  31    N/A  03  B-D  N/A  N/A  L,A           N/A   N/A
              NAME(C_EXTNATTR)

 >000003  ED   000001         0        B8  01-N/A C    N/A  31    N/A  03  B-I  N/A  N/A  L,R,A         N/A   N/A
              NAME(C_CODE)

 >000004  LD   000003         0    000006  01-L   N/A MIN  N/A    N/A  N/A N-I  N/A   S   N             X   00000000
              NAME( )

 >000005  ED   000001         0         0  03-N/A M    N/A  31    N/A  03  B-D  N/A  N/A  L,A,D         N/A   N/A
              NAME(C_WSA)

 >000006  PR   000005     N/A      000018  03-L   N/A N/A  N/A    N/A  03  N/A  N/A   S   N             S     N/A
0             SORT KEY: 00000000 (HEX)
              NAME( )

 >000007  LD   000003        50    000006  01-L   N/A MIN  N/A    N/A  N/A N-I  N/A   S   N             X   00000000
0             EXTENDED ATTRIBUTES: ESDID = 000002, OFFSET=        0
              NAME(main)

 >000008  ER   000001     N/A       N/A    01-L   N    N/S  N/A    N/A  N/A N-D  N/A   S   N/S           S   00000000
              NAME(CEESG003)

 >000009  ER   000001     N/A       N/A    01-L   N    N/S  N/A    N/A  N/A N-D  N/A   S   N/S           S   00000000
              NAME(CBCSG003)

 >000010  SD   000000     N/A       N/A     N/A    N/A N/S  N/A  RENT  N/A N/A  N/A  N/A  N/A           N/A   N/A
              NAME(CEESTART)

 >000011  ED   000010         0        7C  01-N/A C    N/A  31    N/A  03  B-I  N/A  N/A  L,R,A         N/A   N/A
              NAME(C_CODE)

 >000012  LD   000011         0    000000  01-L   N/A MIN  N/A    N/A  N/A N-I  N/A   S   N/S           S   00000000
              NAME(CEESTART)

 >000013  ER   000010     N/A       N/A    01-L   N    N/S  N/A    N/A  N/A N-D  N/A   W   N/S           S   00000000
              NAME(CEEMAIN)

 >000014  ER   000010     N/A       N/A    01-L   N    N/S  N/A    N/A  N/A N-D  N/A   W   N/S           S   00000000
              NAME(CEEFMAIN)

 >000015  ER   000010     N/A       N/A    01-L   N    N/S  N/A    N/A  N/A N-D  N/A   S   N/S           S   00000000
                     ***** G E N E R A L I Z E D  O B J E C T  F I L E  F O R M A T *****              PAGE        2

              NAME(CEEBETBL)

 >000016  ER   000010     N/A       N/A    01-L   N    N/S  N/A    N/A  N/A N-D  N/A   S   N/S           S   00000000
              NAME(CEEBLLST)

 >000017  ER   000010     N/A       N/A    01-L   N    N/S  N/A    N/A  N/A N-I  N/A   S   N/S           S   00000000
              NAME(CEEROOTD)

 >000018  ER   000001     N/A       N/A    01-L   N    N/S  N/A    N/A  N/A N-I  N/A   S   N/S           S   00000000
              NAME(CEESTART)
 >000019  ED   000001         0         0  03-N/A M    N/A  31    N/A  03  B-D  N/A  N/A  L,A           N/A   N/A
              NAME(C_@@PPA2)
```

Figure 15-6. LISTOBJ Format for GOFF (Part 1 of 3)

```
 >000020  PR    000019     N/A      000008  03-L   N/A N/A  N/A   N/A  00  N/A  N/A   S   N           S     N/A
0               SORT KEY: 00000000 (HEX)
                NAME( )

 >000021  ER    000001     N/A       N/A    01-X   N   N/S  N/A   N/A  N/A N-I  N/A   S   G,N         X   00000000
0               EXTENDED ATTRIBUTES: ESDID = 000002, OFFSET=       14
                NAME(__ls__7os-amFPCc)

 >000022  ED    000001         0         0  03-N/A M   N/A  31    N/A  03  B-D  N/A  N/A  L,A,D       N/A   N/A
                NAME(C_WSA)

 >000023  PR    000022     N/A      000000  03-X   N/A N/A  N/A   N/A  03  N/A  N/A   S   N           S     N/A
0               SORT KEY: 00000000 (HEX)
                NAME(cout)

 >000024  SD    000000     N/A       N/A    N/A    N/A N/S  N/A   N/S  N/A N/A  N/A  N/A  N/A         N/A   N/A
                NAME(CEEMAIN)

 >000025  ED    000024         0        10  01-N/A C   N/A  31    N/A  03  B-D  N/A  N/A  L,A         N/A   N/A
                NAME(C_DATA)

 >000026  LD    000025         0    000000  01-L   N/A MIN  N/A   N/A  N/A N-D  N/A   S   N/S         S   00000000
                NAME(CEEMAIN)

 >000027  ER    000001     N/A       N/A    01-L   N   N/S  N/A   N/A  N/A N-I  N/A   S   N/S         S   00000000
                NAME(EDCINPL)

 >000028  ER    000001     N/A       N/A    01-L   N   N/S  N/A   N/A  N/A N-I  N/A   S   N           X   00000000
                NAME(main)

 >000029  ED    000001         0       1B0  01-N/A C   N/A  31    N/A  03  B-D  N/A  N/A  A           N/A   N/A
                NAME(C_COPTIONS)

 >000030  ED    000001         0        22  01-N/A C   N/A  31    N/A  03  F-U  N/A  N/A              N/A   N/A
                NAME(B_IDRL)

 RECORD TYPE: TEXT     SEQUENCE:       32      4
  -- RESIDENT --     TRUE      TEXT    ENCODED
  ESDID   OFFSET    LENGTH   ENCODING  LENGTH    ------------------------------ T E X T ------------------------------

 >000002 00000000  00000000    0000   00000028   00000014 00010001 00010010 00040000 01000000 00000014 00010001 00010010
                                                 00040000 01000000

 >000003 00000000  00000000    0000   000000A0   41F0F050 07FF0700 00000000 F2F0F0F0 F0F1F3F1 F0F8F4F6 F1F6F0F2 F0F9F0F0

1                       ***** G E N E R A L I Z E D  O B J E C T  F I L E  F O R M A T *****              PAGE       3

                                                 02CE07F8 00000080 00000201 00000502 00000038 01000000 00049481 89950000
                                                 00C300C5 00C500F1 FFFFFFE0 00000050 905C47B4 A74AFFB0 0D8047F0 80205860
                                                 48045810 600C1826 98566010 0D764700 00044130 000047F0 802447F0 8004987C
                                                 480C4140 405007F7 FFFFFFB8 01000000 00C300C5 00C500F3 FFFFFFF6 00000000
 >000003 000000A0  00000000    0000   00000018   03012204 FFFFFF60 00000000 FFFFFF6C FFFFFFB0 01000000

 >000006 00000000  00000000    0000   00000018   C8859393 9640E696 99938400 00000000 00000000 00000000

 >000011 00000000  00000000    0000   0000007C   47000000 47000002 90ECD00C 053047F0 30180014 CE030209 0000002C C3C5C5E2
                                                 E3C1D9E3 000058F0 306A050F 00000000 00000000 00000000 00000000 00000000
                                                 FFFE004C 00000000 00000000 00000000 00000000 00000000 00000000 00000000
                                                 00000012 00000000 00000000 00000000 00000000 00000000 00000000

 >000020 00000000  00000000    0000   00000008   00000000 000000A0

 >000025 00000000  00000000    0000   00000010   02000001 00000000 00000000 00000000
```

Figure 15-6. LISTOBJ Format for GOFF (Part 2 of 3)

```
>000029 00000000  00000000    0000   000001B0   C1C7C7D9 C3D6D7E8 4DD5D6D6 E5C5D9D3 C1D75D40 C1D5E2C9 C1D3C9C1 E240C1D9
                                                C3C84DF2 5D40C1D9 C7D7C1D9 E2C540D5 D6C3D6D4 D7D9C5E2 E240D5D6 C3D6D5E5
                                                D3C9E340 D5D6C3E2 C5C3E340 C3E5C6E3 40C4D3D3 4DD5D6C3 C1D3D3C2 C1C3D2C1
                                                D5E85D40 C5E7C5C3 D6D7E240 D5D6C5E7 D7D6D9E3 C1D3D340 C6D3D6C1 E34DC8C5
                                                E76B40C6 D6D3C46B 40D5D6C1 C6D75D40 C7D6C6C6 40D5D6C7 D6D5E4D4 C2C5D940
                                                D5D6C9C7 D5C5D9D9 D5D640D5 D6C9D5C9 E3C1E4E3 D640D5D6 C9D7C140 D3C1D5C7
                                                D3E5D34D C5E7E3C5 D5C4C5C4 5D40D5D6 D3C9C2C1 D5E2C940 D5D6D3D6 C3C1D3C5
                                                40D3D6D5 C7D5C1D4 C540D5D6 D6D7E3C9 D4C9E9C5 40D7D3C9 E2E34DC8 D6E2E35D
                                                40D9C5C4 C9D940D5 D6D9D6C3 D6D5E2E3 40D5D6D9 D6E2E3D9 C9D5C740 D9D6E4D5
                                                C44DE95D 40D5D6E2 C5D9E5C9 C3C540D5 D6E2D6D4 40E2D6D4 C5C9D5C9 E340D5D6
                                                E2D6D4C7 E240E2D7 C9D3D34D F1F2F85D 40E2E3C1 D9E340E2 E3D9C9C3 E340D5D6
                                                E2E3D9C9 C3E36DC9 D5C4E4C3 E3C9D6D5 40E3C1D9 C7C5E34D D3C56B40 D6E2E5F2
                                                D9F95D40 D5D6E3C5 E2E34DC8 D6D6D25D 40E3E4D5 C54DF35D 40E7D7D3 C9D5D240
                                                C3D6D4D7 C9D3C5C4 6DD6D56D D4E5E2FF

RECORD TYPE: IDRL     SEQUENCE:       40        5
  ESDID            |---------- IDR  DATA ---------|    |---------- IDR  DATA ---------|

>000030           |5647A01...02092000031084616000

RECORD TYPE: RLD      SEQUENCE:       41        6
   R-PTR  P-PTR  OFFSET  TYPE LEN ATTRIB   R-PTR  P-PTR  OFFSET  TYPE LEN ATTRIB   R-PTR  P-PTR  OFFSET  TYPE LEN ATTRIB

>000012 000011 000018   00+ 004         000012 000011 000060   00+ 004         000015 000011 000074   00+ 004

>000013 000011 00002C   00+ 004         000014 000011 000068   00+ 004         000016 000011 00006C   00+ 004

>000017 000011 000078   00+ 004         000018 000003 0000A4   00+ 004         000004 000003 0000A4   00- 004

>000004 000020 000004   00+ 004         000021 000006 000014   00+ 004         000021 000006 000010   70+ 004

>000023 000006 00000C   00+ 004         000028 000025 000004   00+ 004         000027 000025 000008   00+ 004

>000028 000025 00000C   70+ 004

RECORD TYPE: END      SEQUENCE:       42         7
 RECORD       --ENTRY POINT--
 COUNT        ESDID    OFFSET
1                      ***** G E N E R A L I Z E D  O B J E C T  F I L E  F O R M A T *****             PAGE       4

>000042       000011  00000000

------------------------------------------------------------------------------------------------------------------------
1                           **  LONG NAME TABLE LISTING OF MEMBER HELLOW    **                           PAGE      1

ABBREVIATION        LONG NAME

__ls__7os-amFPCc := __ls__7ostreamFPCc
                            **  END OF LONG NAME TABLE LISTING OF MEMBER HELLOW    **
```

Figure 15-6. LISTOBJ Format for GOFF (Part 3 of 3)

Display elements in the above figure are described as follows. The numbers enclosed in braces following the field heading are the location (byte.bit) in the GOFF record where the data element can be found.

1 Page Header

– The page header is printed at the top of each page.

– The second line contains an optional user title.

2 HDR Record

This is the first record in each GOFF module. The only data elements printed are the character set identifier and name and the language product (compiler or assembler) which produced the module.

Data elements formatted:
– CCSID
– Character Set Name
– Language Product Identifier

Data elements not formatted:
– Target Hardware Environment (U)
– Target Operating System Environment (U)

3 ESD Record

The ESD describes each external name defined or referenced in the module. Unlike the traditional object module, which provides for up to three names per record, the GOFF format contains only one name per record.

Data elements formatted:

– Line 1
  - ESDID. The identifier for the name being defined or referenced.
  - ESD TYPE. Symbol Type (SD, ED, LD, PR, ER)
  - OWNER/PARENT. The ESDID of the owning or referenced record type in the ESD hierarchy.
  - ITEM OFFSET. The offset, in bytes, of the start of this named entity from the start of the higher level entity.
  - ITEM LENGTH/ADA. For ED- or PR-type ESD records, the length (in bytes) of the entity being defined. If the length field is -1, the true length will be in a LEN-type GOFF record. For LD records, the ESDID of the associated data.
  - NAME SP/S. Name space (00-99) and binding scope (S (local or section),M (module), L (library), or X (import/export)).
  - BA. Binding algorithm (C=Catenate, M=Merge)
  - AMD. AMODE (N/S, 24, 31, ANY, MIN)
  - RMD. RMODE (N/S, 24, 31)
  - REUS. Reusability or tasking behavior. (N/S, NONE, REUS, RENT, REFR)
  - AL. Alignment. Print as decimal value n, where alignment boundary is 2**n. Range: 0-31.
  - TXT. Text type. Displayed in format x-y where
    - x is text record style (B (Byte oriented = 0), F (Fixed = 1), or V (Variable = 2)).
    - y is Executable (U (Unspecified= 0), D (Data=1), I (Instructions=2), or digits 3-7).
  - STR. Binding strength. (S (Strong=0), W (Weak=1)).
  - BINDER. Binder attributes is a string consisting of zero or more of the following characters. The ESD types to which the attribute is applicable are listed in parenthesis.
    - L. Initial or deferred load (ED)
    - M. Movable (ED)
    - R. Read-only (ED)
    - A. Addressable. Text may contain adcons. (ED)
    - C. Common (ED
    - I. Symbol defines or references a descriptor. (LD, ER, PR)
    - G. Mangled name (LD, PR, ER)
    - N. Name may not be renamed. (LD, PR, ER)
    - D. Deferred load (ED)
    - LNK. Linkage Type (S (standard, non-XPLINK), X (XPLINK))
  - SIGNATURE. Any eight-byte string, printed in hexadecimal.

– Sort Key. (Priority) Optional Field. PR only.

– Extended Attributes Optional Field. Defines text location containing additional attributes for this ESD.

– Symbol name. The first line begins with NAME, followed immediately by the name (up to 16 bytes). Names longer than 16 bytes will be abbreviated and displayed here, and an abbreviation-to-long name equivalence table will be listed at the end of the listing. A closing parenthesis will immediately follow the last byte. A name consisting of a single blank character will be displayed as ″NAME( )″.

Data elements not formatted:
– Extended Attribute ESDID (U)
– Extended Attribute Data Offset (U)
– Alias or Alternate Symbol ID (U)
– Name Length (S)

4 TEXT Record

TEXT records are a subset of the TXT record type. They contain the instructions and data of the program. TEXT is displayed in hexadecimal format.

Data elements formatted:

– Line 1
  - ESDID. Identifies the *element* or *part* to which the text belongs.
  - OFFSET. The offset within the element or part where the text is to begin.
  - TRUE LENGTH. The expanded length of the text once the encoding rules (if any) have been applied.
  - TEXT ENCODING. The technique for encoding or decoding the text. Current values are 0 and 1.
  - ENCODED LENGTH. The unexpanded length of the text appearing in this record.
  - TEXT. The text, displayed as it appears in the record. The length of the text to be displayed appears in the ENCODED LENGTH field. Text is displayed in hexadecimal format, 32 bytes per line.

– Lines 2-n

  All text beyond byte 31 is displayed on continuation lines. All bytes beyond the last text byte should be set to blank characters.

Data elements not formatted:
– Data Length (S)

5 IDRL Record

The IDRL provides identification information for the language translator which produced the GOFF. It is a subset of the TXT record type, identified as structured record data. IDRL records will be displayed in 19-byte segments, four per line.

6 RLD Record

The relocation dictionary is a directory of address constants and other data areas which must be modified during binding and loading. Multiple such data areas or adcons can be described in a single RLD record. Relocation directory items begin at {8.0} in the RLD record and vary in length according to the presence or absence of various pointers and offsets in the item.

Directory items are formatted three per line. Each item consists of up to five fields. Flags in the first byte of each directory item indicate which fields are present in the item. As a result, except for the flag bytes in positions 1-8, offsets are not fixed within the directory item as it appears in the GOFF file.

Data elements formatted:

– R-PTR. The ESDID of the target element, the value which will be used in relocating the address constant.

– P-PTR. The ESDID of the element containing the adcon or data area to be modified.
– OFFSET. The offset within the element described by the P-PTR at which the adcon or data area is located.
– TYPE. This describes the type of adcon and implies the operation to be performed on it. Bytes 1 and 2 should be printed in hexadecimal.
– LEN. Length of the adcon or data area. Range: 2-255.
– ATTRIB. Only one attribute is defined in this release. ″H″ should be printed if the addressing mode sensitivity bit is set, meaning that the high order bit of the target field is sensitive to the target AMODE.

Data elements not formatted:
– Total data length (S)
– Flag bytes 0 (except for 0.7) and 5 (S)
– Extended Attributes ESDID and offset (U)

7 END Record

The END record is the last record in the module. It contains a count of the records in the module and an optional entry point nomination, the latter specified by name or by class and offset.

Data elements formatted:

– Line 1
  - RECORD COUNT. Count of the *logical* records in the module, including the HDR and END records.
  - ENTRY POINT ESDID. The identifier of the element containing the entry point.
  - ENTRY POINT OFFSET. The offset of the entry point within the element identified by ESDID.
– Lines 2 contain the symbol name, if specified. The display format is identical to that on the ESD record type.

LISTLOAD OUTPUT=MODLIST Output

```
          LISTLOAD DDN=PDSE,MEMBER=HELLOW,OUTPUT=MODLIST                   00153199
                               ***** M O D U L E   S U M M A R Y *****
   MEMBER NAME:  HELLOW                                                                      MAIN ENTRY POINT:    000000B8
   LIBRARY:      PDSE                                                                        AMODE OF MAIN ENTRY POINT: 31
       NO ALIASES **
------------------------------------------------------------------------------------------------------------------------
                       ****         ATTRIBUTES OF MODULE          ****
                **   BIT  STATUS        BIT  STATUS        BIT  STATUS        BIT  STATUS   **
                      0  NOT-RENT        1  NOT-REUS        2  NOT-OVLY        3  NOT-TEST
                      4  NOT-OL          5  BLOCK           6  EXEC            7  MULTI-RCD
                      8  NOT-DC          9  ZERO-ORG       10 RESERVED        11 RLD
                     12 EDIT            13 NO-SYMS         14 RESERVED        15 NOT-REFR
                     16 RESERVED        17 <16M            18 NOT-PL          19 NO-SSI
                     20 NOT-APF         21 PGM OBJ         22 RESERVED        23 RESERVED
                     24 NOT-ALTP        25 RESERVED        26 RESERVED        27 RMODEANY
                     28 RESERVED        29 RESERVED        30 RESERVED        31 RESERVED
                     32 NON-MIGR        33 NO-PRIME        34 NO-PACK         35 RESERVED
                     36 RESERVED        37 RESERVED        38 RESERVED        39 RESERVED
-----------------------------------------------------------------------------------------------------------------------
                                    MODULE SSI:  NONE
                                    APFCODE:     00000000
                                    RMODE:       ANY
                                    PO FORMAT:   3
                                    XPLINK:      YES
                   *****PROGRAM OBJECT PROCESSED BY BINDER
***THE FOLLOWING ARE THE UNFORMATTED PDSE DIRECTORY ENTRY SECTIONS (PMAR AND PMARL)
PMAR  001E0308 02C00412 00000000 02900000   00B80000 00B80000 00000000 0000
PMARL 00629040 00000000 00050000 02780000   10000000 0B540000 40F40000 00740000
      01400000 00240000 011C0000 00050000   01B40001 00000000 10000000 00000000
      00002001 072F0144 340FD7D4 F6E3C5E2   E3403000 00010000 00180000 20000000
      0178
                                          LISTING OF PROGRAM OBJECT HELLOW                                 PAGE      1
 THIS PROGRAM OBJECT WAS ORIGINALLY PRODUCED BY 5695DF108 AT LEVEL 02.10 ON 03/13/2001 AT 14:43:40
------------------------------------------------------------------------------------------------------------------------
            MODULE SECTION:   $SUMMARY
 USABILITY: UNSPECIFIED    OVERLAY SEGMENT:       0     OVERLAY REGION:       0
 ===== ESDs =====
 C_WSA(ED)
     CLASS:            C_WSA      LENGTH:          18 (HEX)       CLASS OFFSET:            0 (HEX)       FORMAT:  F(0001)
     NAME SPACE:           3      ALIGNMENT:    DOUBLE WORD       BIND METHOD:               MERGE       RMODE:       ANY
     TEXT                         DEFER                           FILL:                     UNSPEC
 C_@@PPA2(ED)
     CLASS:         C_@@PPA2      LENGTH:           8 (HEX)       CLASS OFFSET:            0 (HEX)       FORMAT:  F(0001)
     NAME SPACE:           3      ALIGNMENT:    DOUBLE WORD       BIND METHOD:               MERGE       RMODE:       ANY
     TEXT                         LOAD                            FILL:                     UNSPEC
 $PRIV000013(PD)
     CLASS:         C_@@PPA2      LENGTH:           8 (HEX)       CLASS OFFSET:            0 (HEX)
     NAME SPACE:           3      ALIGNMENT:           BYTE       PRIORITY:                0 (HEX)       SCOPE:    UNSPEC
     ATTRIBUTES:     GENERATED,STRONG
 $PRIV000012(PD)
     CLASS:            C_WSA      LENGTH:          18 (HEX)       CLASS OFFSET:            0 (HEX)
     NAME SPACE:           3      ALIGNMENT:    DOUBLE WORD       PRIORITY:                0 (HEX)       SCOPE:    UNSPEC
     ATTRIBUTES:     GENERATED,STRONG
 ===== TEXT =====            CLASS:            C_WSA
 00000000  C8859393 9640E696 99938400 00000000 00000000 00000000                 Hello.World.....................*
 ===== TEXT =====            CLASS:         C_@@PPA2
 00000000  00000000 000000A0                                                     ................................*
------------------------------------------------------------------------------------------------------------------------
            CONTROL SECTION:  $PRIV000010
 USABILITY: REENTRANT      OVERLAY SEGMENT:       0     OVERLAY REGION:       0
 ===== IDRL =====
           TRANSLATOR   VER   MOD     DATE
           5647A01       02    09    01/31/2000
 ===== ESDs =====
 C_EXTNATTR(ED)
     CLASS:       C_EXTNATTR      LENGTH:          28 (HEX)       CLASS OFFSET:            0 (HEX)       FORMAT:  F(0001)
     NAME SPACE:           1      ALIGNMENT:    DOUBLE WORD       BIND METHOD:            CATENATE       RMODE:       ANY
     TEXT                         LOAD                            FILL:                     UNSPEC
 C_CODE(ED)
     CLASS:           C_CODE      LENGTH:          B8 (HEX)       CLASS OFFSET:            0 (HEX)       FORMAT:  F(0001)
     NAME SPACE:           1      ALIGNMENT:    DOUBLE WORD       BIND METHOD:            CATENATE       RMODE:       ANY
     TEXT                         LOAD                            READ-ONLY                 UNSPEC
 $PRIV000011(LD)
     CLASS:           C_CODE      TEXT TYPE:          INSTR       CLASS OFFSET:            0 (HEX)
```

Figure 15-7. Sample Output for LISTLOAD OUTPUT=MODLIST, ADATA=YES for a Program Object (Part 1 of 6)

```
1                                  LISTING OF PROGRAM OBJECT HELLOW                               PAGE      2

           CONTROL SECTION:  $PRIV000010
===== ESDs =====
    NAME SPACE:          1      SCOPE:           LIBRARY      ELEMENT OFFSET:         0 (HEX)      AMODE:        ANY
    ATTRIBUTES:     XPL,STRONG,MAPPED                         ADA:            $PRIV000012
C_WSA(ED)
    CLASS:            C_WSA      LENGTH:           0 (HEX)      CLASS OFFSET:           0 (HEX)      FORMAT:  F(0001)
    NAME SPACE:          3      ALIGNMENT:    DOUBLE WORD      BIND METHOD:             MERGE      RMODE:        ANY
    TEXT                        DEFER                         FILL:                   UNSPEC
$PRIV000012(PR)
    CLASS:            C_WSA      LENGTH:          18 (HEX)      CLASS OFFSET:           0 (HEX)
    NAME SPACE:          3      ALIGNMENT:    DOUBLE WORD      PRIORITY:               0 (HEX)      SCOPE:    UNSPEC
    ATTRIBUTES:     STRONG,MAPPED
main(LD)
    CLASS:           C_CODE      TEXT TYPE:          INSTR      CLASS OFFSET:          50 (HEX)
    NAME SPACE:          1      SCOPE:           LIBRARY      ELEMENT OFFSET:        50 (HEX)      AMODE:        ANY
    ATTRIBUTES:     XPL,STRONG,MAPPED                         ADA:            $PRIV000012
    EXTENDED ATTRIBUTES: CLASS=C_EXTNATTR, OFFSET=000000
CEESG003(ER)
                                TEXT TYPE:           DATA      CLASS OFFSET:           0 (HEX)
    TARGET SECTION:                                           TARGET CLASS:
    NAME SPACE:          1      SCOPE:           LIBRARY      ELEMENT OFFSET:         0 (HEX)
    UNRESOLVED                  AUTOCALL
    ATTRIBUTES:     STRONG
CBCSG003(ER)
                                TEXT TYPE:           DATA      CLASS OFFSET:           0 (HEX)
    TARGET SECTION:                                           TARGET CLASS:
    NAME SPACE:          1      SCOPE:           LIBRARY      ELEMENT OFFSET:         0 (HEX)
    UNRESOLVED                  AUTOCALL
    ATTRIBUTES:     STRONG
CEESTART(ER)
                                TEXT TYPE:          INSTR      CLASS OFFSET:          B8 (HEX)
    TARGET SECTION: CEESTART                                  TARGET CLASS:            C_CODE
    NAME SPACE:          1      SCOPE:           LIBRARY      ELEMENT OFFSET:         0 (HEX)
    RESOLVED                    AUTOCALL
    ATTRIBUTES:     STRONG
C_@@PPA2(ED)
    CLASS:         C_@@PPA2      LENGTH:           0 (HEX)      CLASS OFFSET:           0 (HEX)      FORMAT:  F(0001)
    NAME SPACE:          3      ALIGNMENT:    DOUBLE WORD      BIND METHOD:             MERGE      RMODE:        ANY
    TEXT                        LOAD                          FILL:                   UNSPEC
$PRIV000013(PR)
    CLASS:         C_@@PPA2      LENGTH:           8 (HEX)      CLASS OFFSET:           0 (HEX)
    NAME SPACE:          3      ALIGNMENT:           BYTE      PRIORITY:               0 (HEX)      SCOPE:    UNSPEC
    ATTRIBUTES:     STRONG,MAPPED
__ls__7os-amFPCc(ER)
                                TEXT TYPE:          INSTR      CLASS OFFSET:           0 (HEX)
    TARGET SECTION:                                           TARGET CLASS:
    NAME SPACE:          1      SCOPE:           EXP/IMP      ELEMENT OFFSET:         0 (HEX)
    UNRESOLVED                  AUTOCALL
    ATTRIBUTES:     XPL,STRONG,MAPPED,MANGLED
    EXTENDED ATTRIBUTES: CLASS=C_EXTNATTR, OFFSET=000014
C_WSA(ED)
    CLASS:            C_WSA      LENGTH:           0 (HEX)      CLASS OFFSET:           0 (HEX)      FORMAT:  F(0001)
    NAME SPACE:          3      ALIGNMENT:    DOUBLE WORD      BIND METHOD:             MERGE      RMODE:        ANY
    TEXT                        DEFER                         FILL:                   UNSPEC
```

Figure 15-7. Sample Output for LISTLOAD OUTPUT=MODLIST, ADATA=YES for a Program Object (Part 2 of 6)

```
1                                           LISTING OF PROGRAM OBJECT HELLOW                                 PAGE      3

            CONTROL SECTION:  $PRIV000010
===== ESDs =====
cout(PR)
    CLASS:           C_WSA      LENGTH:          0 (HEX)       CLASS OFFSET:          0 (HEX)
    NAME SPACE:          3      ALIGNMENT:    DOUBLE WORD      PRIORITY:              0 (HEX)       SCOPE:   EXP/IMP
    ATTRIBUTES:     STRONG,MAPPED
EDCINPL(ER)
                                TEXT TYPE:         INSTR       CLASS OFFSET:          0 (HEX)
    TARGET SECTION:                                            TARGET CLASS:
    NAME SPACE:          1      SCOPE:           LIBRARY       ELEMENT OFFSET:        0 (HEX)
    UNRESOLVED                  AUTOCALL
    ATTRIBUTES:     STRONG
main(ER)
                                TEXT TYPE:         INSTR       CLASS OFFSET:         50 (HEX)
    TARGET SECTION: $PRIV000010                                TARGET CLASS:           C_CODE
    NAME SPACE:          1      SCOPE:           LIBRARY       ELEMENT OFFSET:       50 (HEX)
    RESOLVED                    AUTOCALL
    ATTRIBUTES:     XPL,STRONG,MAPPED
C_COPTIONS(ED)
    CLASS:      C_COPTIONS      LENGTH:        1B0 (HEX)       CLASS OFFSET:          0 (HEX)       FORMAT:  F(0001)
    NAME SPACE:          1      ALIGNMENT:    DOUBLE WORD      BIND METHOD:          CATENATE       RMODE:       ANY
    TEXT                        NOLOAD                         FILL:                   UNSPEC
===== RLDs =====            CLASS:         C_@@PPA2
ELEM.OFF CLS.OFF  TYPE STATUS LENG HOBCHG NSPACE      TARGET NAME      PARTRES          XPL  XATTR NAME      XATTR OFF
00000004 00000004 N-BR   RES  0004   NO      1     (+)$PRIV000011      $PRIV000013        NO
===== RLDs =====            CLASS:            C_WSA
ELEM.OFF CLS.OFF  TYPE STATUS LENG HOBCHG NSPACE      TARGET NAME      PARTRES          XPL  XATTR NAME      XATTR OFF
0000000C 0000000C N-BR UNRES  0004   NO      3     (+)cout             $PRIV000012        NO
00000010 00000010 DATA UNRES  0004   NO      1     (+)__ls__7os-amFPCc $PRIV000012        NO  C_EXTNATTR       000014
00000014 00000014   BR UNRES  0004   NO      1     (+)__ls__7os-amFPCc $PRIV000012        NO  C_EXTNATTR       000014
===== RLDs =====            CLASS:           C_CODE
ELEM.OFF CLS.OFF  TYPE STATUS LENG HOBCHG NSPACE      TARGET NAME      PARTRES          XPL  XATTR NAME      XATTR OFF
000000A4 000000A4 N-BR   RES  0004   NO      1     (+)CEESTART                            NO
000000A4 000000A4 N-BR   RES  0004   NO      1     (+)$PRIV000011                         NO
===== TEXT =====            CLASS:       C_EXTNATTR
00000000  00000014 00010001 00010010 00040000 01000000 00000014 00010001 00010010   *................................*
00000020  00040000 01000000                                                          ................................*
===== TEXT =====            CLASS:           C_CODE
00000000  41F0F050 07FF0700 00000000 F2F0F0F0 F0F1F3F1 F0F8F4F6 F1F6F0F2 F0F9F0F0   *.00.........20000131084616020900*
00000020  02CE07F8 00000080 00000201 00000502 00000038 01000000 00049481 89950000   *...8.......................main..*
00000040  00C300C5 00C500F1 FFFFFFE0 00000050 905C47B4 A74AFFB0 0D8047F0 80205860   *.C.E.E.1............x......0....*
```

Figure 15-7. Sample Output for LISTLOAD OUTPUT=MODLIST, ADATA=YES for a Program Object (Part 3 of 6)

```
1                                        LISTING OF PROGRAM OBJECT HELLOW                                      PAGE      4

0           CONTROL SECTION:  $PRIV000010
 ===== TEXT =====            CLASS:           C_CODE
000000060  48045810 600C1826 98566010 0D764700 00044130 000047F0 802447F0 8004987C   *........q..............0...0..q.*
 00000080  480C4140 405007F7 FFFFFFB8 01000000 00C300C5 00C500F3 FFFFFFF6 00000000   *.......7.........C.E.E.3...6....*
 000000A0  03012204 00000018 00000000 FFFFFF6C FFFFFFB0 01000000                       ................................*
 ===== TEXT =====            CLASS:       C_COPTIONS
000000000  C1C7C7D9 C3D6D7E8 4DD5D6D6 E5C5D9D3 C1D75D40 C1D5E2C9 C1D3C9C1 E240C1D9   *AGGRCOPY.NOOVERLAP..ANSIALIAS.AR*
 00000020  C3C84DF2 5D40C1D9 C7D7C1D9 E2C540D5 D6C3D6D4 D7D9C5E2 E240D5D6 C3D6D5E5   *CH.2..ARGPARSE.NOCOMPRESS.NOCONV*
 00000040  D3C9E340 D5D6C3E2 C5C3E340 C3E5C6E3 40C4D3D3 4DD5D6C3 C1D3D3C2 C1C3D2C1   *LIT.NOCSECT.CVFT.DLL.NOCALLBACKA*
 00000060  D5E85D40 C5E7C5C3 D6D7E240 D5D6C5E7 D7D6D9E3 C1D3D340 C6D3D6C1 E34DC8C5   *NY..EXECOPS.NOEXPORTALL.FLOAT.HE*
 00000080  E76B40C6 D6D3C46B 40D5D6C1 C6D75D40 C7D6C6C6 40D5D6C7 D6D5E4D4 C2C5D940   *X..FOLD..NOAFP..GOFF.NOGONUMBER.*
 000000A0  D5D6C9C7 D5C5D9D9 D5D640D5 D6C9D5C9 E3C1E4E3 D640D5D6 C9D7C140 D3C1D5C7   *NOIGNERRNO.NOINITAUTO.NOIPA.LANG*
 000000C0  D3E5D34D C5E7E3C5 D5C4C5C4 5D40D5D6 D3C9C2C1 D5E2C940 D5D6D3D6 C3C1D3C5   *LVL.EXTENDED..NOLIBANSI.NOLOCALE*
 000000E0  40D3D6D5 C7D5C1D4 C540D5D6 D6D7E3C9 D4C9E9C5 40D7D3C9 E2E34DC8 D6E2E35D   *.LONGNAME.NOOPTIMIZE.PLIST.HOST.*
 00000100  40D9C5C4 C9D940D5 D6D9D6C3 D6D5E2E3 40D5D6D9 D6E2E3D9 C9D5C740 D9D6E4D5   *.REDIR.NOROCONST.NOROSTRING.ROUN*
 00000120  C44DE95D 40D5D6E2 C5D9E5C9 C3C540D5 D6E2D6D4 40E2D6D4 C5C9D5C9 E340D5D6   *D.Z..NOSERVICE.NOSOM.SOMEINIT.NO*
 00000140  E2D6D4C7 E240E2D7 C9D3D34D F1F2F85D 40E2E3C1 D9E340E2 E3D9C9C3 E340D5D6   *SOMGS.SPILL.128..START.STRICT.NO*
 00000160  E2E3D9C9 C3E36DC9 D5C4E4C3 E3C9D6D5 40E3C1D9 C7C5E34D D3C56B40 D6E2E5F2   *STRICT.INDUCTION.TARGET.LE..OSV2*
 00000180  D9F95D40 D5D6E3C5 E2E34DC8 D6D6D25D 40E3E4D5 C54DF35D 40E7D7D3 C9D5D240   *R9..NOTEST.HOOK..TUNE.3..XPLINK.*
 000001A0  C3D6D4D7 C9D3C5C4 6DD6D56D D4E5E2FF                                         COMPILED.ON.MVS.................*
0-------------------------------------------------------------------------------------------------------------------------
0           CONTROL SECTION:  CEEMAIN
0USABILITY: UNSPECIFIED    OVERLAY SEGMENT:       0     OVERLAY REGION:       0
 ===== ESDs =====
0C_DATA(ED)
     CLASS:           C_DATA      LENGTH:          10 (HEX)       CLASS OFFSET:          0 (HEX)       FORMAT:  F(0001)
     NAME SPACE:           1      ALIGNMENT:    DOUBLE WORD       BIND METHOD:          CATENATE       RMODE:       ANY
     TEXT                         LOAD                            FILL:                   UNSPEC
 CEEMAIN(LD)
     CLASS:           C_DATA      TEXT TYPE:           DATA       CLASS OFFSET:          0 (HEX)
     NAME SPACE:           1      SCOPE:            LIBRARY       ELEMENT OFFSET:        0 (HEX)       AMODE:       ANY
     ATTRIBUTES:     STRONG
 ===== RLDs =====            CLASS:           C_DATA
0ELEM.OFF CLS.OFF  TYPE STATUS LENG HOBCHG NSPACE      TARGET NAME      PARTRES           XPL  XATTR NAME      XATTR OFF
000000004 00000004   BR   RES  0004   NO      1     (+)main                                NO
 00000008 00000008   BR UNRES  0004   NO      1     (+)EDCINPL                             NO
 0000000C 0000000C DATA UNRES  0004   NO      1     (+)main                                NO
 0000000C 0000000C C-OF   RES  0004   NO      3     (+)$PRIV000012                         NO
 ===== TEXT =====            CLASS:           C_DATA
000000000  02000001 00000050 00000000 00000000                                         ................................*
0-------------------------------------------------------------------------------------------------------------------------
```

Figure 15-7. Sample Output for LISTLOAD OUTPUT=MODLIST, ADATA=YES for a Program Object (Part 4 of 6)

```
1                                           LISTING OF PROGRAM OBJECT HELLOW                                  PAGE      5

0           CONTROL SECTION:  CEESTART
0USABILITY: REENTRANT      OVERLAY SEGMENT:       0     OVERLAY REGION:       0
 ===== ESDs =====
0C_CODE(ED)
     CLASS:           C_CODE      LENGTH:          7C (HEX)       CLASS OFFSET:          B8 (HEX)       FORMAT:  F(0001)
     NAME SPACE:           1      ALIGNMENT:    DOUBLE WORD       BIND METHOD:           CATENATE       RMODE:       ANY
     TEXT                         LOAD                            READ-ONLY                UNSPEC
 CEESTART(LD)
     CLASS:           C_CODE      TEXT TYPE:          INSTR       CLASS OFFSET:          B8 (HEX)
     NAME SPACE:           1      SCOPE:            LIBRARY       ELEMENT OFFSET:         0 (HEX)       AMODE:       ANY
     ATTRIBUTES:     STRONG
 CEEMAIN(ER)
                                  TEXT TYPE:           DATA       CLASS OFFSET:           0 (HEX)
     TARGET SECTION: CEEMAIN                                      TARGET CLASS:            C_DATA
     NAME SPACE:           1      SCOPE:            LIBRARY       ELEMENT OFFSET:         0 (HEX)
     RESOLVED                     AUTOCALL
     ATTRIBUTES:     WEAK
 CEEFMAIN(ER)
                                  TEXT TYPE:           DATA       CLASS OFFSET:           0 (HEX)
     TARGET SECTION:                                              TARGET CLASS:
     NAME SPACE:           1      SCOPE:            LIBRARY       ELEMENT OFFSET:         0 (HEX)
     UNRESOLVED                   AUTOCALL
     ATTRIBUTES:     WEAK
 CEEBETBL(ER)
                                  TEXT TYPE:           DATA       CLASS OFFSET:           0 (HEX)
     TARGET SECTION:                                              TARGET CLASS:
     NAME SPACE:           1      SCOPE:            LIBRARY       ELEMENT OFFSET:         0 (HEX)
     UNRESOLVED                   AUTOCALL
     ATTRIBUTES:     STRONG
 CEEBLLST(ER)
                                  TEXT TYPE:           DATA       CLASS OFFSET:           0 (HEX)
     TARGET SECTION:                                              TARGET CLASS:
     NAME SPACE:           1      SCOPE:            LIBRARY       ELEMENT OFFSET:         0 (HEX)
     UNRESOLVED                   AUTOCALL
     ATTRIBUTES:     STRONG
 CEEROOTD(ER)
                                  TEXT TYPE:          INSTR       CLASS OFFSET:           0 (HEX)
     TARGET SECTION:                                              TARGET CLASS:
     NAME SPACE:           1      SCOPE:            LIBRARY       ELEMENT OFFSET:         0 (HEX)
     UNRESOLVED                   AUTOCALL
     ATTRIBUTES:     STRONG
 ===== RLDs =====            CLASS:          C_CODE
0ELEM.OFF CLS.OFF  TYPE STATUS LENG HOBCHG NSPACE      TARGET NAME      PARTRES           XPL  XATTR NAME      XATTR OFF
000000018 000000D0 N-BR   RES  0004   NO      1     (+)CEESTART                            NO
 0000002C 000000E4   BR   RES  0004   NO      1     (+)CEEMAIN                             NO
 00000060 00000118 N-BR   RES  0004   NO      1     (+)CEESTART                            NO
 00000068 00000120   BR UNRES  0004   NO      1     (+)CEEFMAIN                            NO
 0000006C 00000124   BR UNRES  0004   NO      1     (+)CEEBLLST                            NO
 00000074 0000012C   BR UNRES  0004   NO      1     (+)CEEBETBL                            NO
 00000078 00000130   BR UNRES  0004   NO      1     (+)CEEROOTD                            NO
 ===== TEXT =====            CLASS:          C_CODE
0000000B8  47000000 47000002 90ECD00C 053047F0 30180014 CE030209 000000E4 C3C5C5E2   *...............0...........UCEES*
```

Figure 15-7. Sample Output for LISTLOAD OUTPUT=MODLIST, ADATA=YES for a Program Object (Part 5 of 6)

```
1                                           LISTING OF PROGRAM OBJECT HELLOW                                 PAGE      6

0            CONTROL SECTION:  CEESTART
 ===== TEXT =====             CLASS:            C_CODE
0000000D8  E3C1D9E3 000058F0 306A050F 00000000 00000000 00000000 00000000 00000000   *TART...0........................*
 000000F8  FFFE004C 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*
 00000118  000000CA 00000000 00000000 00000000 00000000 00000000 00000000             ................................*
0-------------------------------------------------------------------------------------------------------------------------
0            CONTROL SECTION:  IEWBLIT
0USABILITY: UNSPECIFIED    OVERLAY SEGMENT:       0     OVERLAY REGION:       0
 ===== ESDs =====
0B_LIT(ED)
     CLASS:            B_LIT      LENGTH:         100 (HEX)       CLASS OFFSET:           0 (HEX)       FORMAT:  F(0001)
     NAME SPACE:           1      ALIGNMENT:    DOUBLE WORD       BIND METHOD:          CATENATE        RMODE:       ANY
     TEXT                         LOAD                            FILL:                   UNSPEC
 IEWBLIT(LD)
     CLASS:            B_LIT      TEXT TYPE:            DATA      CLASS OFFSET:           0 (HEX)
     NAME SPACE:           1      SCOPE:              MODULE      ELEMENT OFFSET:         0 (HEX)       AMODE:       ANY
     ATTRIBUTES:     GENERATED,STRONG
 ===== RLDs =====             CLASS:            B_LIT
0ELEM.OFF CLS.OFF  TYPE STATUS LENG HOBCHG NSPACE       TARGET NAME      PARTRES           XPL  XATTR NAME      XATTR OFF
000000028 00000028 LTKN   RES  0008   NO      0     (+)                                    NO
 00000050 00000050  CPR   RES  0004   NO      0        C_CODE                              NO
 00000054 00000054 SEGM   RES  0004   NO      0     (+)C_CODE                              NO
 00000070 00000070  CPR   RES  0004   NO      0        C_EXTNATTR                          NO
 00000074 00000074 SEGM   RES  0004   NO      0     (+)C_EXTNATTR                          NO
 00000090 00000090  CPR   RES  0004   NO      0        C_@@PPA2                            NO
 00000094 00000094 SEGM   RES  0004   NO      0     (+)C_@@PPA2                            NO
 000000B0 000000B0  CPR   RES  0004   NO      0        C_DATA                              NO
 000000B4 000000B4 SEGM   RES  0004   NO      0     (+)C_DATA                              NO
 000000D0 000000D0  CPR   RES  0004   NO      0        C_WSA                               NO
 000000F0 000000F0  CPR   RES  0004   NO      0        B_LIT                               NO
 000000F4 000000F4 SEGM   RES  0004   NO      0     (+)B_LIT                               NO
 ===== TEXT =====             CLASS:            B_LIT
000000000  C9C5E6C2 D3C9E340 00000100 01000000 00000040 00000020 00000006 00000001   *IEWBLIT.........................*
 00000020  00000000 00000000 00000000 00000000 00000000 80000000 00000000 00000000   *................................*
 00000040  C36DC3D6 C4C54040 40404040 40404040 00000134 00000000 03038000 00000000   *C.CODE..........................*
 00000060  C36DC5E7 E3D5C1E3 E3D94040 40404040 00000028 00000000 03030000 00000000   *C.EXTNATTR......................*
 00000080  C36D7C7C D7D7C1F2 40404040 40404040 00000008 00000000 03030000 00000000   *C...PPA2........................*
 000000A0  C36DC4C1 E3C14040 40404040 40404040 00000010 00000000 03030000 00000000   *C.DATA..........................*
 000000C0  C36DE6E2 C1404040 40404040 40404040 00000018 00000000 03032000 00000000   *C.WSA...........................*
 000000E0  C26DD3C9 E3404040 40404040 40404040 00000100 00000000 03030000 00000000   *B.LIT...........................*
------------------------------------------------------------------------------------------------------------------------
 ==== MERGE CLASS PART INITIALIZERS ====
 CLASS            PART             OFFSET REPEAT ---------------------  I N I T I A L  T E X T  -----------------------

1                                           LISTING OF PROGRAM OBJECT HELLOW                                 PAGE      7

 ==== MERGE CLASS PART INITIALIZERS ====
 CLASS            PART             OFFSET REPEAT ---------------------  I N I T I A L  T E X T  -----------------------
 C_WSA            $PRIV000012      000000 00001  00000000 40404040 40404040 40404040 40404040 40404040 40404040 40404040
 C_@@PPA2         $PRIV000013      000000 00001  00000000 40404040 40404040 40404040 40404040 40404040 40404040 40404040
------------------------------------------------------------------------------------------------------------------------
1                             **  LONG NAME TABLE LISTING OF PROGRAM OBJECT HELLOW           **            PAGE      8

 ABBREVIATION        LONG NAME

 __ls__7os-amFPCc := __ls__7ostreamFPCc
                             **  END OF LONG NAME TABLE LISTING OF PROGRAM OBJECT HELLOW          **
**   END OF PROGRAM OBJECT LISTING
```

Figure 15-7. Sample Output for LISTLOAD OUTPUT=MODLIST, ADATA=YES for a Program Object (Part 6 of 6)

Description of MODLIST Output for a Program Object

The listing produced by LISTLOAD OUTPUT=MODLIST consists of multiple parts (see Figure 15-7 on page 15-28):

A page heading, displayed at the top of each page.

The page heading consists of one or two heading lines, in the following format:

```
LISTING OF PROGRAM OBJECT xxxxxxxx
```

The heading lines are followed by the title line, entered in the TITLE parameter of the LISTLOAD control statement.

The binder-generated program object identification record (IDRB).

The IDRB record is displayed on a line by itself, in the format:

```
THIS PROGRAM OBJECT WAS ORIGINALLY PRODUCED BY 5695DF108 AT LEVEL 02.10
ON mm/dd/yyyy AT hh:mm:ss
```

The binder program identifier, version and level, date and time of binding are presented here. The IDRB line is followed by a line of dashes.

An individual listing for each control section in the program object, separated by a dashed line.

For each control section in the module, the ESD Section Definition (SD) record is formatted, followed by all data classes in the following sequence:

1. IDRZ - SPZAP identification data
2. IDRL - Language translator identification data
3. IDRU - User-supplied identification data
4. SYM - internal symbol dictionary
5. ESD - External Symbol Dictionary
6. RLD - ReLocation Dictionary
7. TEXT - Instructions and data for the CSECT
8. ADATA - ADATA information

The SD record occupies two print lines:

- The first begins with one of the constants CONTROL SECTION, SEGMENT TABLE, ENTRY TABLE, or MODULE SECTION, and displays either the section or common name. If there is no user-defined name, then a binder-generated name will be displayed as follows:
  - $PRIV*xxxxx*, where *x* is a number for user sections which originally had blank names or were unnamed.
  - $BLANKCOM - unnamed common $SUMMARY binder-generated section containing merge classes for the module
- The second line displays the USABILITY, the overlay segment and region. USABILITY must contain one of the values UNSPECIFIED, NON-REUSABLE, REUSABLE, REENTRANT or REFRESHABLE. For non-overlay modules, the latter two fields should contain zero.

Each of the eight class subsections begin with an identifier line of the format:

```
===== class name =====
```

IDR detail is in the same format as described in “Description of LISTIDR Output” on page 15-48, except that it is displayed only for the single section. The remainder of the classes are described below:

- SYM data is displayed 40 bytes per line
- ESD data occupies three to four lines per ESD record. The first containing the external name (abbreviated name, if name longer than 16 bytes), followed by the ESD record type in parenthesis. The rest of the formatted fields vary depending on the ESD record type:
  - ED records define an element definition. Its length, and various attributes will be used to bind and load the class contained in the section. Each ED record occupies three print lines:
    1. The first line displays:
       - CLASS name - up to 16 bytes.
       - LENGTH of defined class element in hexadecimal.
       - CLASS OFFSET - in hexadecimal.

    - FORMAT - where the first field is the class record format with either F (fixed length record), or V (variable length record), follows by the hexadecimal value of the record format in parenthesis.
  2. The second line shows:
    - NAME SPACE - in hexadecimal.
    - ALIGNMENT - DOUBLE WORD, QUAD WORD, or PAGE.
    - BIND METHOD - CATENATE or MERGE.
    - RMODE - 24, ANY, or UNSPECIFIED
  3. The third line displays the Binder and Loader attributes:
    - Binder attributes can be DESCRIPTIVE DATA, or TEXT
    - Loader attributes can be NOLOAD, LOAD, DEFER, READ-ONLY (or FILL: UNSPEC is printed, if there is no fill character).
- ER records define external references from the named section. Each ER record occupies four print lines, where:
  1. The first line displays:
    - TEXT TYPE - can be either UNSPEC (unspecified), INSTRUC (instructions or code), DATA, or TRANS.DEF (translator defined).
    - CLASS OFFSET - in hexadecimal.
  2. The second line shows:
    - TARGET SECTION - Target section name (abbreviated name, if name is longer than 16 bytes).
    - TARGET CLASS - Name of class containing target label.
  3. The third line shows:
    - NAME SPACE - in hexadecimal.
    - SCOPE - Scope of name (SECTION, MODULE, LIBRARY, or IMP/EXP).
    - ELEMENT OFFSET - in hexadecimal.
  4. The fourth line displays the ER status and autocall, where:
    - ER status can either be RESOLVED or UNRESOLVED
    - AUTOCALL can either be AUTOCALL or NEVERCALL
  5. The fifth line displays binder attributes. Possible attributes are XPL, STRONG or WEAK, MAPPED, INDIRECT, or GENERATED.
- LD records define a label or entry point in the named section. Each LD record occupies three print lines, where:
  1. The first line displays:
    - CLASS NAME - up to 16 bytes.
    - TEXT TYPE - can be either UNSPEC, INSTR, DATA, or TRANS.DEF
    - CLASS OFFSET - in hexadecimal.
  2. The second line shows:
    - NAME SPACE - in hexadecimal.
    - SCOPE - Scope of name (SECTION, MODULE, LIBRARY, or IMP/EXP).
    - ELEMENT OFFSET - in hexadecimal.
    - AMODE - can either be 24, 31, MIN, ANY, or UNSPECIFIED
  3. The third line displays the binder attributes. Possible attributes are XPL (xplink), STRONG or WEAK, MAPPED, INDIRECT, or

GENERATED (the LD record was generated by the binder). In addition, if the record contains the name of the symbol which defines the environment or associated data (ADA), that symbol will be printed.

4. If extended attributes exist, a fourth line will contain the resident class and offset.

– PD (Part Definition) and PR (Part Reference) records define parts or pseudo registers. The PR record is a local definition of the part (within the section), whereas the PD record is a global definition for all of the associated PRs (PRs with the same name). PR and PD records contain the same formatted fields. Each record occupies three print lines, where:
  1. The first line displays:
     - CLASS name - up to 16 bytes.
     - LENGTH - in hexadecimal.
     - CLASS OFFSET - in hexadecimal.
  2. The second line shows:
     - NAME SPACE - in hexadecimal.
     - ALIGNMENT - in either BYTE, HALF WORD, FULL WORD, DOUBLE WORD, or PAGE.
     - PRIORITY - Controls the order of the part within the element.
     - SCOPE - Scope of part (SECTION, MODULE, LIBRARY, or IMP/EXP).
  3. The third line displays binder attributes. Possible attributes are: XPL, STRONG or WEAK, MAPPED, INDIRECT, or GENERATED.

- RLD data is displayed one line per RLD record, by element offset of the associated address constant. Where multiple RLD records refer to the same adcon, the element offsets will be the same. RLD data shown consists of:
  – Element Offset - The offset, in hex, of the associated address constant within the element.
  – Class Offset - The offset, in hex, of the associated address constant within the class.
  – ADCON TYPE - The type of address constant associated with this RLD entry. Five types are supported: BR (V-type), N-BR (A-type), SEGM (address of class segment), C-OF (Q-type), or CPR (cumulative class length).
  – Status - This field identifies the status of the associated address constant. Valid status values are: RES (resolved), UNRES (unresolved), and N-REL (non-relocatable constant).
  – LENGTH - The adcon length in hexadecimal.
  – HOBCHG - High order bit of V-type adcon was changed by the Binder. Possible value can be either YES or NO.
  – NAME SPACE of reference in hexadecimal.
  – TARGET NAME - Name of the referenced symbol.
  – PARTRES - If the RLD describes an adcon on a part (PR), this will be the name of the resident part.
  – XATTR NAME - Symbol defining the location at which the extended attributes (if any) are stored.
  – XATTR OFF - Offset from symbol XATTR NAME at which the extended attributes are stored.
- TEXT data is displayed by class name. In addition to the hexadecimal representation, text data is in EBCDIC format.

- ADATA information, if requested via ADATA=YES on the LISTLOAD OUTPUT=MODLIST control card, like TEXT data is displayed by class name in both hexadecimal and EBCDIC presentation.

  The abbreviation-to-long name equivalence table is displayed prior to end of the listing, with all the abbreviated names (external names exceeding 16 bytes in length) in the formatted part of the listing with their long names.

A trailer record.

```
** END OF PROGRAM OBJECT LISTING
```

LISTLOAD OUTPUT=XREF Output

This section shows three samples of cross-reference listings:

- Figure 15-8 on page 15-38 shows the output from a program object version 2, that contains multiple classes. See the descriptions following this figure for explanations.
- Figure 15-9 on page 15-44 and Figure 15-10 on page 15-45 allow you to compare the output for a load module with the output for a program object version 1.

The listing has the following parts:
- Numerical map, which presents information approximately in the order in which it appears in the module.
- Numerical cross-reference
- Alphabetical map, which presents information alphabetically by symbol name.
- Alphabetical cross-reference

In the following listing, page 1 shows the numerical map of the module. Page 2 shows the numerical cross-reference list of the module. Page 4 shows the alphabetical map, and page 5 the alphabetical cross-reference list.

Note: The module shown in the Figure 15-8 on page 15-38 is not in overlay format; for overlay modules, each segment is formatted separately.

As with the other output from AMBLIST, each page begins with a standard heading. The first line of each page contains a page number and begins with one of the following heading constants:
- NUMERICAL MAP OF PROGRAM OBJECT ....
- NUMERICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT ....
- ALPHABETICAL MAP OF PROGRAM OBJECT ....
- ALPHABETICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT ....

The member name will appear following the heading. If the name is more than sixteen characters, its formatted 16-bytes abbreviation name is printed instead. An optional second line will be used to print the title information, provided by the user on the LISTLOAD control statement. Each of the four parts has its own subheading line(s), to describe the detail that follows.

```
A    ** NUMERICAL MAP OF PROGRAM OBJECT LOADMOD1         **                        PAGE      1


------------------------------------------------------------------------------------------------------------------------
CLASS NAME:            B_PRV
      CLAS LOC   ELEM LOC    LENGTH  TYPE    ALIGNMENT             NAME
        0                         0   ED     BYTE                  $PRIVATE
             0                    1   PD     BYTE                    A
             1                    1   PD     BYTE                    E
             2                    1   PD     BYTE                    C
             3                    1   PD     BYTE                    G
             4                    2   PD     HALF WORD               F
             8                    4   PD     FULL WORD               D
            10                    8   PD     DOUBLE WORD             B
        0                         0   ED     BYTE                  SD1
      CLASS LENGTH                0
------------------------------------------------------------------------------------------------------------------------
B     ** NUMERICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT LOADMOD1         **       PAGE      2

         CLAS LOC   ELEM LOC           CLASS NAME
ADCON AT       C6        C6                                  IN SECTION  SD1    1
             $CLASS_OFFSET                   B_PRV             REFERS TO  A  2
 ADCON AT      12E       12E                                  IN SECTION  SD1
             $CLASS_OFFSET                   B_PRV             REFERS TO  B
ADCON AT      196       196                                  IN SECTION  SD1
             $CLASS_OFFSET                   B_PRV             REFERS TO  C
ADCON AT      1FE       1FE                                  IN SECTION  SD1
             $CLASS_OFFSET                   B_PRV             REFERS TO  D
ADCON AT      266       266                                  IN SECTION  SD1
             $CLASS_OFFSET                   B_PRV             REFERS TO  E
ADCON AT      2CE       2CE                                  IN SECTION  SD1
             $CLASS_OFFSET                   B_PRV             REFERS TO  F
ADCON AT      336       336                                  IN SECTION  SD1
             $CLASS_OFFSET                   B_PRV             REFERS TO  G
ADCON AT      39C       39C                                  IN SECTION  SD1
             $CLASS_LEN                      B_PRV             REFERS TO  $CLASS_LEN
------------------------------------------------------------------------------------------------------------------------
CLASS NAME:           B_TEXT
      CLAS LOC   ELEM LOC    LENGTH  TYPE    ALIGNMENT             NAME
        0                       430   ED     DOUBLE WORD           SD1
      430                       238   ED     DOUBLE WORD           SD2
      668                         8   ED     DOUBLE WORD           SDX
           668          0             LD                             LD1
           66C          4             LD                             LD2
     1000                        30   ED     DOUBLE WORD           $BLANKCOM     3
     2000                        30   ED     DOUBLE WORD           CM1
      CLASS LENGTH                0
------------------------------------------------------------------------------------------------------------------------
```

Figure 15-8. Sample Output for LISTLOAD OUTPUT=XREF for a Program Object with Class Names: B_PRV and B_TEXT (Part 1 of 3)

```
          ** NUMERICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT LOADMOD1         **       PAGE      3

         CLAS LOC   ELEM LOC          CLASS NAME
ADCON AT       48       48                                IN SECTION  SD1
                0        0               B_TEXT             REFERS TO  SD1
ADCON AT      478       48                                IN SECTION  SD2
              430        0               B_TEXT             REFERS TO  SD2
ADCON AT      4F4       C4                                IN SECTION  SD2
              66C        4               B_TEXT             REFERS TO  LD2
              668        0                                    IN SECTION  SDX
ADCON AT      554      124                                IN SECTION  SD2
             2000        0               B_TEXT             REFERS TO  CM1
ADCON AT      604      1D4                                IN SECTION  SD2
             2000        0               B_TEXT             REFERS TO  CM1
LENGTH OF PROGRAM OBJECT     2030
------------------------------------------------------------------------------------------------------------------------
C   **  ALPHABETICAL MAP OF PROGRAM OBJECT LOADMOD1         **                      PAGE      4

ENTRY NAME  CLAS LOC  ELEM LEN/LOC            CLASS NAME       SECTION NAME OR ENTRY TYPE
$BLANKCOM
               1000        30                      B_TEXT       (ED)
$PRIVATE
                  0         0                       B_PRV       (ED)
A
                  0         1                       B_PRV       (PD)
B
                 10         8                       B_PRV       (PD)
C
                  2         1                       B_PRV       (PD)
CM1
               2000        30                      B_TEXT       (ED)
D
                  8         4                       B_PRV       (PD)
E
                  1         1                       B_PRV       (PD)
F
                  4         2                       B_PRV       (PD)
G
                  3         1                       B_PRV       (PD)
LD1
                668        0                       B_TEXT       SDX
LD2
                66C        4                       B_TEXT       SDX
SDX
                668         8                      B_TEXT       (ED)
```

Figure 15-8. Sample Output for LISTLOAD OUTPUT=XREF for a Program Object with Class Names: B_PRV and B_TEXT (Part 2 of 3)

```
SD1
                 0          0                      B_PRV       (ED)
SD1
                 0        430                     B_TEXT       (ED)
SD2
               430        238                     B_TEXT       (ED)
------------------------------------------------------------------------------------------------------------------------
D     ** ALPHABETICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT LOADMOD1         **    PAGE      5

      CLAS LOC  ELEM LOC             CLASS NAME       SYMBOL
      $CLASS_LEN                          B_PRV       $CLASS_LEN              1
          39C       39C                                   REFERENCED IN  SD1
      $CLASS_OFFSET                       B_PRV       A
           C6        C6                                   REFERENCED IN  SD1
      $CLASS_OFFSET                       B_PRV       B
          12E       12E                                   REFERENCED IN  SD1      3
      $CLASS_OFFSET                       B_PRV       C
          196       196                                   REFERENCED IN  SD1
     2000             0                  B_TEXT       CM1
          554       124                                   REFERENCED IN  SD2
     2000             0                  B_TEXT       CM1
          604       1D4                                   REFERENCED IN  SD2
      $CLASS_OFFSET                       B_PRV       D
          1FE       1FE                                   REFERENCED IN  SD1
      $CLASS_OFFSET                       B_PRV       E
          266       266                                   REFERENCED IN  SD1
      $CLASS_OFFSET                       B_PRV       F
          2CE       2CE                                   REFERENCED IN  SD1
      $CLASS_OFFSET                       B_PRV       G
          336       336                                   REFERENCED IN  SD1
      66C             4                  B_TEXT       LD2
          668         0                                 IN SECTION  SDX             2
          4F4        C4                                   REFERENCED IN  SD2
        0             0                  B_TEXT       SD1
           48        48                                   REFERENCED IN  SD1
      430             0                  B_TEXT       SD2
          478        48                                   REFERENCED IN  SD2
LENGTH OF PROGRAM OBJECT     2030
**   END OF MAP AND CROSS-REFERENCE LISTING
```

Figure 15-8. Sample Output for LISTLOAD OUTPUT=XREF for a Program Object with Class Names: B_PRV and B_TEXT (Part 3 of 3)

Numerical Map

The **A** *Numerical Map* prints one line for each defined element definition, part, or control section in the composite ESD. The detail line contains the class offset (in hex), either a section offset for (labels/parts) or a length (for control sections/ element definitions), the ESD record type, alignment (for LD/PD record type), and the label/part, section or element definition name. Sections generated by the binder, or binder-generated names for unnamed user sections, will be displayed as:

$PRIV*xxxxxx* - where *xxxxxx* is numeric.
$BLANKCOM
$SEGTAB
$ENTAB

All other entries will contain a valid name, assigned by the user for a label/part, a control section/element definition or named common. For label (LD) or part (PD) type ESD entries, the class offset and label/part name will be indented to show that the label/part is contained within the previous section entry.

If the module is in overlay format, the map and cross-reference will alternate for each segment. In this case, the map will begin with a segment identifier line:

```
SEG. nnnnn ----------------------------------------------------
```

and will end with a segment length line:

```
LENGTH OF SEGMENT  nnnnn
```

The numerical map is functionally equivalent to the load map produced by the linkage editor or binder.

Numerical Cross-Reference

The **B** numerical cross-reference listing contains one entry for each RLD record in the module, presented in sequence by the hexadecimal class offset of the related address constant. There is one RLD record for each A-, V-, and Q-type address constant in the module, and one RLD record for each class reference (RLD type=21), class length (CXD), or loader token.

Each entry consists of two or three lines, depending upon the RLD type and resolution status of the reference. Resolved relocatable adcons (A-types and V-types) require three lines, whereas unresolved references, non-relocatable constants (Q-type), and class reference (RLD type=21) require only two. The three lines are described as follows:

1 The first line describes the adcon itself, showing the class and element offsets, in hex, and the name of the section containing the adcon. Because all adcons must reside within a section, there will always be either a user-defined section name or a representation of the binder-generated name, such as $PRIV000001 or $BLANKCOM. **2** The second line describes the referenced, or target, symbol. It contains the class and element offsets of the referenced label/part or section/element definition, and the class name of the referenced class name, if the reference is resolved, or one of the following constants:

- $UNRESOLVED - A strong reference (ER) could not be resolved during binding.
- $UNRESOLVED(W) - A weak reference (WX) could not be resolved during binding.
- $NEVER CALL - The symbol was marked never call, and no attempt was made during binding to resolve the symbol from the library.
- $CLASS-OFFSET - The reference was to a class offset (Q-con).
- $CLASS-LEN - The reference was to a class length (RLD type=40).

The second line also shows the name of the referenced symbol, following the constant string REFERS TO. If the RLD item is for a class offset or class length, the constant string $CLASS_OFFSET or $CLASS_LEN will appear in place of a name.

3 The third line will be printed only for resolved A-type and V-type address constants. It describes the section containing the referent label, and includes module offset (section offset is always zero for sections), and the section name. If the target section does not have a name, then a representation of the binder-generated name ($PRIV000005, $BLANKCOM) will be printed. If the target name in the second line matches the containing section name, the third line will not be printed, since it would provide no additional information over the second line.

The last, or only, segment cross-reference will be followed by the length of the program object:

```
LENGTH OF PROGRAM OBJECT nnnn
```

If no RLD available in a class, the following message will appear instead of the formatted detail:

```
**** NO ADCONS IN THIS CLASS ****
```

Alphabetical Map

The **C** alphabetical map displays label definitions, part definitions, control sections and element definitions (except ER and PR) in alphabetical sequence, two print lines per ESD entry. It contains all of the same information as the Numerical Map, but in a different sequence. This part always begins on a new page, with a standard page heading of ALPHABETICAL MAP OF PROGRAM OBJECT .....

The first detail line contains the label, section or common name.

The second detail line consists of the class offset, the element offset (type LD/PD records) or element definition length (all other types), the class name (name of the containing class), and the name of the containing section/element (type LD/PD records) or the ESD entry type (all other types). Element lengths are indented, to distinguish them from element offsets. If the module is in overlay format, the segment number is printed to the right of the section length.

Alphabetical Cross-Reference

The **D** alphabetical cross-reference listing provides the same information as the numerical cross-reference listing, but in a different sequence. This part of the report is in collating sequence by referenced name (the name of the symbol being referred to in the address constant).

The alphabetical cross-reference begins on a new page with a standard page heading ALPHABETICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT, and contains four columns (only three of which appear in the non-overlay example):

1. Class offset. This is the hex offset of the named item within a class of the program object. Class offsets for the second and third detail lines have been indented.
2. Element offset. This is the hex offset of the named label or part within its section. Lines referring to an element, rather than a label, will always display zero for the element offset.
3. Overlay segment. This is displayed for overlay format modules only.
4. Symbol. This field varies between the three detail lines, as described in the following text. If the displayed name is a special section name, then one of the binder-generated names for example, $PRIV*xxxxxx*), described earlier, will replace the name.

As in the numerical cross-reference listing, the alphabetical cross-reference detail is presented in two or three lines. In this case, line 2 is optional and appears only for adcons that have been resolved to labels (as opposed to sections). The three lines are described as follows:

1 Line 1 describes the referenced (target) symbol, and displays the class offset, element offset or zero, optionally the segment number (if overlay), the referent class name, and symbol name. If the reference is unresolved, then the offset field will be overlaid with the constant $UNRESOLVED - A strong reference (ER) could not be resolved during binding. If the RLD is of type 30 (class offset), or type 40 (class len), the offset fields will be overlaid with one of the following constants:
$CLASS_OFFSET - The reference was to a class offset.
$CLASS_LEN - The reference was to a class length.

2 Line 2 only appears for adcons that have been resolved to labels (LD type ESD entries). It describes the section containing the referent label, including the class offset, a zero element offset, optional segment number, and section name. The section name is preceded by the constant IN SECTION, which is indented from the name field in line 1.

3 Line 3 describes the referencing adcon, including its class and element offsets, an optional segment number, and the name of the section containing the adcon. The section name is preceded by the constant REFERENCED IN, which is indented from the name field in the preceding line.

If no RLD is available in a program object, the following message will appear instead of the formatted detail:

```
**** NO RLD DATA ***
```

The cross reference listing concludes with the line

```
**   END OF MAP AND CROSS-REFERENCE LISTING
```

LISTLOAD OUTPUT=XREF Output (Comparison of Load Module and Program Object Version 1)

```
    LISTLOAD OUTPUT=XREF,DDN=DD1,
        MEMBER=MAINRTN
                              *****  M O D U L E   S U M M A R Y  *****
    MEMBER NAME:  MAINRTN                                                          MAIN ENTRY POINT:    00000000
    LIBRARY:      DD1                                                              AMODE OF MAIN ENTRY POINT: 31
        NO ALIASES **
------------------------------------------------------------------------------------------------------------------------
                      ****   LINKAGE EDITOR ATTRIBUTES OF MODULE   ****
                **   BIT  STATUS        BIT  STATUS        BIT  STATUS        BIT  STATUS   **
                      0  NOT-RENT        1  NOT-REUS        2  NOT-OVLY        3  NOT-TEST
                      4  NOT-OL          5  BLOCK           6  NOT-EXEC        7  MULTI-RCD
                      8  NOT-DC          9  ZERO-ORG        10 EP-ZERO         11 RLD
                      12 EDIT            13 NO-SYMS         14 F-LEVEL         15 NOT-REFR
------------------------------------------------------------------------------------------------------------------------
                                    MODULE SSI:  NONE
                                    APFCODE:     00000000
                                    RMODE:       ANY
                   *****LOAD MODULE PROCESSED BY VS LINKAGE EDITOR
                             NUMERICAL MAP AND CROSS-REFERENCE LIST OF LOAD MODULE MAINRTN                   PAGE 0001

                 CONTROL SECTION                                          ENTRY
                  LMOD LOC     NAME      LENGTH  TYPE                      LMOD LOC  CSECT LOC      NAME
                       00    MAINRTN        166   SD
                      168   $PRIVATE         BC   PC
                                                                                1D4       6C      NONAME1M
                      228   $PRIVATE         BC   PC
                                                                                294       6C      NONAME2M
                      2E8    SUBRTN         102   SD
                      3F0    AAAAAAAA        54   CM
                      448   $BLANKCOM        54   CM
------------------------------------------------------------------------------------------------------------------------
      LMOD LOC  CSECT LOC    IN CSECT           REFERS TO SYMBOL  AT LMOD LOC  CSECT LOC    IN CSECT
           AC        AC      MAINRTN                AAAAAAAA           3F0        00      AAAAAAAA
           B0        B0      MAINRTN                NONAME1M           1D4        6C     $PRIVATE
           B4        B4      MAINRTN                NONAME2M           294        6C     $PRIVATE
          160       160      MAINRTN                SUBRTN             2E8        00      SUBRTN
          394        AC      SUBRTN                $BLANKCOM           448        00     $BLANKCOM
LENGTH OF LOAD MODULE     4A0
                                        ALPHABETICAL MAP OF LOAD MODULE MAINRTN                                PAGE 0002

           CONTROL SECTION                                          ENTRY
               NAME     LMOD LOC   LENGTH  TYPE                        NAME    LMOD LOC  CSECT LOC  CSECT NAME
            $BLANKCOM      448        54   CM
            $PRIVATE       168        BC   PC
            $PRIVATE       228        BC   PC
             AAAAAAAA      3F0        54   CM
             MAINRTN        00       166   SD
                                                                      NONAME1M      1D4      6C       $PRIVATE
                                                                      NONAME2M      294      6C       $PRIVATE
             SUBRTN        2E8       102   SD
                               ALPHABETICAL CROSS-REFERENCE LIST OF LOAD MODULE MAINRTN                     PAGE 0003

  SYMBOL   AT LMOD LOC  CSECT LOC    IN CSECT           IS REFERRED TO BY LMOD LOC  CSECT LOC    IN CSECT
$BLANKCOM       448         00     $BLANKCOM                          394              AC      SUBRTN
 AAAAAAAA       3F0         00      AAAAAAAA                           AC              AC      MAINRTN
 NONAME1M       1D4         6C     $PRIVATE                            B0              B0      MAINRTN
 NONAME2M       294         6C     $PRIVATE                            B4              B4      MAINRTN
 SUBRTN         2E8         00      SUBRTN                            160             160      MAINRTN
**    END OF MAP AND CROSS-REFERENCE LISTING
```

Figure 15-9. Sample Output for LISTLOAD OUTPUT=XREF for a Load Module

```
  LISTLOAD OUTPUT=XREF,DDN=DD1,
      MEMBER=(MAINRTN,
 THISISALONGALIASNAMEYOUMAYCHANGETHENAMEIFYOULIKEANYNAMEWILLDOOOO),
      TITLE=('XREF LISTINGS OF A LONG ALIAS NAME',10)
                             ***** M O D U L E   S U M M A R Y *****
    MEMBER NAME:  MAINRTN                                                            MAIN ENTRY POINT:    00000000
    LIBRARY:      DD1                                                                AMODE OF MAIN ENTRY POINT: 31
        ** ALIASES **                                                      ALIAS ENTRY POINT    AMODE OF ALIAS ENTRY POINT
          THISISALONGALIASNAMEYOUMAYCHANGETHENAMEIFYOULIKEANYNAMEWILLDOOOO    00000000                 31
------------------------------------------------------------------------------------------------------------------------
                      ****         ATTRIBUTES OF MODULE          ****
               **  BIT  STATUS        BIT  STATUS        BIT  STATUS        BIT  STATUS   **
                    0  NOT-RENT        1  NOT-REUS        2  NOT-OVLY        3  NOT-TEST
                    4  NOT-OL          5  BLOCK           6  EXEC            7  MULTI-RCD
                    8  NOT-DC          9  ZERO-ORG       10 RESERVED        11 RLD
                   12 EDIT            13 NO-SYMS         14 RESERVED        15 NOT-REFR
                   16 RESERVED        17 <16M            18 NOT-PL          19 NO-SSI
                   20 NOT-APF         21 PGM OBJ         22 RESERVED        23 RESERVED
                   24 RESERVED        25 RESERVED        26 RESERVED        27 RMODEANY
------------------------------------------------------------------------------------------------------------------------
                                    MODULE SSI:  NONE
                                    APFCODE:     00000000
                                    RMODE:       ANY
                 *****PROGRAM OBJECT PROCESSED BY BINDER
THIS PROGRAM OBJECT WAS ORIGINALLY PRODUCED BY 5695DF108 AT LEVEL 01.00 ON 09/16/92 AT 09:11:52
------------------------------------------------------------------------------------------------------------------------
                                NUMERICAL MAP AND CROSS-REFERENCE LIST OF PROGRAM OBJECT MAINRTN           PAGE      1
         XREF LISTINGS OF A LONG ALIAS NAME
      LMOD LOC   SECT LOC    LENGTH  TYPE           NAME
       0                       168   SD            MAINRTN
      168                       58   CM            AAAAAAAA
      1C0                       C0   PC            $PRIVATE
         22C        6C                               NONAME1M
      280                       C0   PC            $PRIVATE
         2EC        6C                               NONAME2M
      340                      108   SD            SUBRTN
      448                       58   CM            $BLANKCOM
------------------------------------------------------------------------------------------------------------------------
         LMOD LOC   SECT LOC
ADCON AT       AC        AC          IN SECTION  MAINRTN
              168         0            REFERS TO  AAAAAAAA
ADCON AT       B0        B0          IN SECTION  MAINRTN
              22C        6C            REFERS TO  NONAME1M
              1C0         0              IN SECTION  $PRIVATE
ADCON AT       B4        B4          IN SECTION  MAINRTN
              2EC        6C            REFERS TO  NONAME2M
              280         0              IN SECTION  $PRIVATE
ADCON AT      160       160          IN SECTION  MAINRTN
              340         0            REFERS TO  SUBRTN
ADCON AT      3EC        AC          IN SECTION  SUBRTN
              448         0            REFERS TO  $BLANKCOM
LENGTH OF PROGRAM OBJECT      4A0
```

Figure 15-10. Sample Output for LISTLOAD OUTPUT=XREF for a Program Object (Part 1 of 2)

```
                                          ALPHABETICAL MAP OF PROGRAM OBJECT MAINRTN                     PAGE      2
         XREF LISTINGS OF A LONG ALIAS NAME
ENTRY NAME  LMOD LOC  SECT LEN/LOC            SECTION NAME OR ENTRY TYPE
$BLANKCOM
                448        58                 (CM)
$PRIVATE
                1C0        C0                 (PC)
$PRIVATE
                280        C0                 (PC)
AAAAAAAA
                168        58                 (CM)
MAINRTN
                  0       168                 (SD)
NONAME1M
                22C            6C             $PRIVATE
NONAME2M
                2EC            6C             $PRIVATE
SUBRTN
                340       108                 (SD)
------------------------------------------------------------------------------------------------------------------------
                                   ALPHABETICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT MAINRTN             PAGE      3
         XREF LISTINGS OF A LONG ALIAS NAME
      LMOD LOC  SECT LOC            SYMBOL
      448            0              $BLANKCOM
          3EC       AC                  REFERENCED IN  SUBRTN
      168            0              AAAAAAAA
           AC       AC                  REFERENCED IN  MAINRTN
      22C           6C              NONAME1M
          1C0        0                IN SECTION  $PRIVATE
           B0       B0                  REFERENCED IN  MAINRTN
      2EC           6C              NONAME2M
          280        0                IN SECTION  $PRIVATE
           B4       B4                  REFERENCED IN  MAINRTN
      340            0              SUBRTN
          160      160                  REFERENCED IN  MAINRTN
**   END OF MAP AND CROSS-REFERENCE LISTING
```

Figure 15-10. Sample Output for LISTLOAD OUTPUT=XREF for a Program Object (Part 2 of 2)

LISTIDR Output

```
  LISTIDR  DDN=DD1,MEMBER=TLIST
                                  ***** M O D U L E   S U M M A R Y *****
    MEMBER NAME:  TLIST                                                                                 MAIN ENTRY POINT:    0000000E
    LIBRARY:      DD1                                                                                   AMODE OF MAIN ENTRY POINT: 24
        NO ALIASES **
------------------------------------------------------------------------------------------------------------------------
                       ****          ATTRIBUTES OF MODULE           ****
                **   BIT  STATUS         BIT  STATUS         BIT  STATUS         BIT  STATUS   **
                      0  NOT-RENT         1  NOT-REUS         2  NOT-OVLY         3  NOT-TEST
                      4  NOT-OL           5  BLOCK            6  EXEC             7  MULTI-RCD
                      8  NOT-DC           9  ZERO-ORG        10 EP > ZERO        11 RLD
                     12 EDIT             13 NO-SYMS          14 F-LEVEL          15 NOT-REFR
------------------------------------------------------------------------------------------------------------------------
                                    MODULE SSI:  NONE
                                    APFCODE:     00000000
                                    RMODE:       24
                   *****LOAD MODULE PROCESSED EITHER BY VS LINKAGE EDITOR OR BINDER
                                           LISTIDR FOR LOAD MODULE TLIST                       PAGE 0001

B                   CSECT                        YR/DAY                      SPZAP    DATA
                    A                            1972/271                      92240
                    B                            1972/271                      NO IDENT
------------------------------------------------------------------------------------------------------------------------

A           THIS LOAD MODULE WAS PRODUCED BY LINKAGE EDITOR 5695DF108  AT LEVEL 21.01 ON DAY 271 OF YEAR 1992.
------------------------------------------------------------------------------------------------------------------------

C         CSECT        TRANSLATOR             VR.MD                           YR/DY
          A            566896201              02.01                           1972/271
          B            566896201              02.01                           1972/271
          D1           566896201              02.01                           1972/271
          UNRES        566896201              02.01                           1992/034
------------------------------------------------------------------------------------------------------------------------

D         CSECT                  YR/DAY                    USER
          A                      1972/271                    ANOTHERONE
          B                      1972/271                    myprogram
------------------------------------------------------------------------------------------------------------------------
```

Figure 15-11. Sample LISTIDR Output for a Load Module Processed by Linkage Editor or Binder

```
 LISTIDR     MEMBER=(LOADMOD2)                                          00020905
                                  *****  M O D U L E   S U M M A R Y  *****
    MEMBER NAME:  LOADMOD2                                                                   MAIN ENTRY POINT:    00000000
    LIBRARY:      SYSLIB                                                                     AMODE OF MAIN ENTRY POINT: 31
        NO ALIASES **
------------------------------------------------------------------------------------------------------------------------
                       ****         ATTRIBUTES OF MODULE           ****
                **   BIT  STATUS        BIT  STATUS        BIT  STATUS        BIT  STATUS   **
                      0  NOT-RENT        1  NOT-REUS        2  NOT-OVLY        3  NOT-TEST
                      4  NOT-OL          5  BLOCK           6  EXEC            7  MULTI-RCD
                      8  NOT-DC          9  ZERO-ORG       10 RESERVED        11 RLD
                     12 EDIT            13 NO-SYMS         14 RESERVED        15 NOT-REFR
                     16 RESERVED        17 <16M            18 NOT-PL          19 NO-SSI
                     20 NOT-APF         21 PGM OBJ         22 RESERVED        23 RESERVED
                     24 NOT-ALTP        25 RESERVED        26 RESERVED        27 RMODEANY
                     28 RESERVED        29 RESERVED        30 RESERVED        31 RESERVED
                     32 MIGRATE         33 NO-PRIME        34 NO-PACK         35 RESERVED
                     36 RESERVED        37 RESERVED        38 RESERVED        39 RESERVED
------------------------------------------------------------------------------------------------------------------------
                                    MODULE SSI:  NONE
                                    APFCODE:     00000000
                                    RMODE:       ANY
                                    CREATED BY:  00000001

A            *****PROGRAM OBJECT PROCESSED BY BINDER
***THE FOLLOWING ARE THE UNFORMATTED PDSE DIRECTORY ENTRY SECTIONS (PMAR AND PMARL)
PMAR  001E0206 02C00412 00000000 04500000   00000000 00000000 00000000 0000
PMARL 005200C0 00000000 00020000 04500000   02380000 08180000 0EF40000 00500000
      01240000 00200000 01040000 00020000   01740001 00000000 04500000 00000000
      00001995 154F0205 408FC2D7 C2C6F6F4   F5F5
                                          LISTIDR FOR PROGRAM OBJECT LOADMOD2                              PAGE      1

------------------------------------------------------------------------------------------------------------------------
THIS PROGRAM OBJECT WAS ORIGINALLY PRODUCED BY 5695DF108 AT LEVEL 01.01 ON 06/03/95 AT 20:54:08
------------------------------------------------------------------------------------------------------------------------
B DATE        PTF NUMBER
CSECT:    SDX
          04/16/2001   ZAPIDR01
          04/16/2001   ZAPIDR02
          04/16/2001   ZAPIDR03
          04/16/2001   ZAPIDR04
          04/16/2001   ZAPIDR05
          04/16/2001   ZAPIDR06
          04/16/2001   ZAPIDR07
          04/16/2001   ZAPIDR08
          04/16/2001   ZAPIDR09
------------------------------------------------------------------------------------------------------------------------
C TRANSLATOR   VER   MOD     DATE
CSECT:    CM1
          566896201     02    01    04/16/2001
CSECT:    SDX
          566896201     02    01    04/16/2001
CSECT:    SD1
          566896201     02    01    04/16/2001
CSECT:    SD2
          566896201     02    01    04/16/2001
------------------------------------------------------------------------------------------------------------------------
D DATE         USER DATA
CSECT:    $MODULE LEVEL DATA
          04/16/2001   THIS IS A TEST
CSECT:    SD1
          04/16/2001   USER IDR TEST 2
          04/16/2001   USER IDR TEST 3
 ------------------------------------------------------------------------------------------------------------------------
```

Figure 15-12. Sample LISTIDR Output for a Program Object Processed by Binder

Description of LISTIDR Output

As shown in Figure 15-11 on page 15-47 and Figure 15-12, the IDR listing has four sections, separated by dashed lines. The four sections contain the following:

A The linkage editor identification or binder identification record (IDRB). The identification record is displayed in a single line. This line shows the binder or linkage editor program identifier, version and release numbers, and the data and time of binding.

Note: The time of binding is listed only for a program object.

B A list of SPZAP IDR entries (IDRZ), if any. The IDRZ records, if any, are formatted two or more lines per section. The first contains the associated CSECT name, and the second, and subsequent lines, a modification date and up to eight bytes of PTF number or other data entered on the SPZAP IDRDATA control statement. There will be one detail line for each modification to the control section. For load module output, the IDRZ records are formatted one line per section.

C A list of language translator IDR records (IDRL). These entries are formatted only if OUTPUT=ALL was specified, or defaulted, on the LISTIDR control statement. The IDRL records, if any, are also formatted two or more lines per CSECT. The section name appears on the first line, and the translator program id, version and release, and date of translation on the second and subsequent lines. There will be one line of translator data for each compiler, assembler or other language product involved in the production of the object code for that section. For load module output, the IDRL records are formatted one line per section.

D A list of user-supplied IDR data (IDRU), if any. The IDRU records normally appear two lines per CSECT. The first line shows the section name, and the second line an entry date and up to 80 bytes of data, entered by the user on the binder IDENTIFY control statement. If the section name is a module level section (identified as ’00000001’x), the constants $MODULE LEVEL DATA are printed in place of the section name.

For program objects, if no data is available in a section, one of the following messages will appear instead of the formatted detail:

```
        NO SPZAP DATA EXISTS FOR THIS PROGRAM OBJECT
       NO BINDER DATA EXISTS FOR THIS PROGRAM OBJECT
  NO TRANSLATION DATA EXISTS FOR THIS PROGRAM OBJECT
NO IDENTITY/USER DATA EXISTS FOR THIS PROGRAM OBJECT
```

For load modules, if no SPZAP data is available, the following message will appear instead of the formatted detail:

```
THIS LOAD MODULE CONTAINS NO INFORMATION SUPPLIED BY SPZAP
```

LISTLPA Output

```
MODIFIED  LINK  PACK  AREA  MAP  -   ALPHABETICALLY BY NAME
NAME      LOCATION  LENGTH    EP ADDR   MAJOR  LPDE  NAME    NAME      LOCATION  LENGTH    EP ADDR   MAJOR  LPDE  NAME
IGC00020  --------  ------    00B42000                       IGC0005E  --------  ------    00B37FA0
IGC0006I  --------  ------    00B419C0                       IGGC019BN --------  ------    00B414D8
                                                                N  00.12SEC  VIRT 200K  SYS  276K
MODIFIED  LINK  PACK  AREA  MAP  -   NUMERICALLY BY ENTRY POINT
NAME      LOCATION  LENGTH    EP ADDR   MAJOR  LPDE  NAME    NAME      LOCATION  LENGTH    EP ADDR   MAJOR  LPDE  NAME
IGC0005E  --------  ------    00B37FA0                       IGGC019BN --------  ------    00B414D8
IGC0006I  --------  ------    00B419C0                       IGC00020  --------  ------    00B42000
                                                                N  00.12SEC  VIRT 200K  SYS  276K
PAGEABLE  LINK  PACK  AREA  MAP  -   ALPHABETICALLY BY NAME
NAME      LOCATION  LENGTH    EP ADDR   MAJOR  LPDE  NAME    NAME      LOCATION  LENGTH    EP ADDR   MAJOR  LPDE  NAME
AHLACFV                       819B595E  AHLTVTAM             AHLDMPMD                      81926EBE  AHLSETD
AHLDSP                        81963962  AHLTXSYS             AHLEXT                        8198F660  AHLTSYSM
AHLFI0                        8193A926  AHLTSYFL             AHLFPI                        8193A9FC  AHLTSYFL
AHLFRR                        8198F7EA  AHLTSYSM             AHLFSSCH                      8193A946  AHLTSYFL
AHLFSVC                       8193A9D8  AHLTSYFL             AHLMCER                       81926450  AHLTSETD
AHLPINT                       8198F748  AHLTSYSM             AHLREADR  01977C08  000003F8  81977C08
AHLSBCU1                      81991F4A  AHLWSMOD             AHLSBL0K                      819916B0  AHLWSMOD
AHLSBUF                       81991A90  AHLWSMOD             AHLSETD   01926000  00001708  81926000
AHLSETEV  01928000  00001998  81928000                       AHLSFE0B                      819917EE  AHLWSMOD
AHLSRB                        819639EE  AHLTXSYS             AHLSRM                        81963A62  AHLTXSYS
AHLSTAE                       8198F8C6  AHLTSYSM             AHLSVC                        8198F61A  AHLTSYSM
AHLTACFV                      819B596A  AHLTVTAM             AHLTCCWG  0192A000  00002378  8191A000
AHLTDIR                       81926A58  AHLSETD              AHLTDSP                       81971658  AHLTPID
AHLTEXT   01956920  000006E0  81956920                       AHLTFCG   0192D000  000016D0  8192D000
AHLTF0R   01954570  00000A90  81954570                       AHLTFRR                       81954694  AHLTFOR
AHLTLSR                       819717D2  AHLTPID              AHLTPI                        8197147E  AHLTPID
AHLTPID   01971468  00000B98  81971468                       AHLTSLIP  0192F000  00001C50  8192F000
AHLTSRB                                 AHLTPID              AHLTSRM                                 AHLTFOR
                              81971770                                                     8195458C
AHLSTAE                                 AHLTFOR              AHLTSVC   01931000  00002768
                              819547B4                                                     81931000
AHLTSYFL  0193A908  000006F8                                 AHLTSYSM  0198F508  00000AF8
                              8193A908                                                     8198F508
AHLTUSR   019299C0  00000640                                 AHLTVTAM  019B5940  000006C0
                              819299C0                                                     819B5940
AHLTXSYS  01963850  000007B0                                 AHLVCOFF  019B6F40  000000C0
                              81963850                                                     819B6F40
AHLVCON   01989EE8  00000118                                 AHLWSMOD  019916B0  00000950
                              81989EE8                                                     819916B0
AHLWTOMD                                AHLSETD              AMDSYS00  01934000  00001208
                              81926E4C                                                     81934000
AMDSYS01  01936000  00002AD8                                 AMDSYS02  019BB648  00000548
                              81936000                                                     819BB648
AMDSYS03  01939000  00001828                                 AMDSYS04  0193B000  00002038
                              91039000                                                     8193B000
          01975178  99999358                                           01961C08  000003F8
                              81975178                                                     81961C08
AMDSYS05  00F28000  00001E60                                 AMDSYS06  00BF1008  000006C8
                              00F28000                                                     00BF1008
AMDUSRFD                                IMDUSRFF             AMDUSRFE                                IMDUSRF8
                              00C4C000                                                     00C08590
AMDUSRFF  00B8E230  000003F8                                 AMDUSRF8                                ISTAICIR
                              00B8E230                                                     00C48000
AMDUSRF9  00C4E730  000008D0                                 CCKRIUWT                                DCM3B3
                              00C4E730                                                     00C56328
CVAFGTF                                 DCM3B3               DCMBE0    00F26000  00001360
                              00C56328                                                     00F26000
DCMBE1    00CB8078  00000F88                                 DCM180    00C54020  00000FE0
                              00CB8078                                                     00C54020
DCM181    00C26318  00000CE8                                 DCM182    00F24000  000014E0
DCM183                        00C26318  DCM270               DCM270    00E1E830  000007D0  00F24000
DCM271                        00F24000                       DCM272                        00E1E830
```

Figure 15-13. Sample LISTLPA Output

LISTLOAD Output: DAT-ON Nucleus

```
                                 LISTING  OF  LOAD  MODULE  PL1LOAD                                  PAGE  0001

RECORD# 1     TYPE  20  -  CESD  ESDID  1                        ESD  SIZE  240

              CESD#    SYMBOL     TYPE     ADDRESS      SEGNUM   ID/LENGTH(DEC)     (HEX)
                 1    PL1TC02    00(SD)     000000        1           1206           4B6
                 2    PL1TC02A   00(SD)     0004B8        1            608           260
                 3    IHEQINV    06(PR)     000000        3              4             4
                 4    IHESADA    02(ER)     000000
                 5    INESADB    02(ER)     000000
                 6    IHEQERR    06(PR)     000004        3              4             4
                 7    IHEQTIC    06(PR)     000008        3              4             4
                 8    IHEMAIN    00(SD)     000718        1              4             4
                 9    IHENTRY    00(SD)     000720        1             12             C
                10    IHESAPC    02(ER)     000000
                11    IHEQLWF    06(PR)     00000C        3              4             4
                12    IHEQSLA    06(PR)     000010        3              4             4
                13    IHEQLW0    06(PR)     000014        3              4             4
                14    PL1TC02B   06(PR)     000018        3              4             4
                15    PL1TC02C   06(PR)     00001C        3              4             4

RECORD# 2     TYPE  20  -  CESD  ESDID  16                       ESD  SIZE  240

              CESD#    SYMBOL     TYPE     ADDRESS      SEGNUM   ID/LENGTH(DEC)     (HEX)
                16    IHELD0A    02(ER)     000000
                17    IHELD0B    02(ER)     000000
                18    IHEI0BT    02(ER)     000000
                19    IHEI0BC    02(ER)     000000
                20    IHESAFA    02(ER)     000000
                21    IHESAFB    02(ER)     000000
                22    AA         02(ER)     000000
                23    C          00(SD)     000730        1              4             4
                24    B          00(SD)     000738        1              4             4
                25    A          00(SD)     000740        1              4             4
                26    IHESPRT    00(SD)     000748        1             56            38
                27    IHEQSPR    06(PR)     000020        3              4             4
                28    IHEDNC     02(ER)     000000
                29    IHEVPF     02(ER)     000000
                30    IHEDMA     02(ER)     000000

RECORD# 3     TYPE  20  -  CESD  ESDID  31                       ESD  SIZE  64

              CESD#    SYMBOL     TYPE     ADDRESS      SEGNUM   ID/LENGTH(DEC)     (HEX)
                31    IHEVPB     02(ER)     000000
                32    IHEVSC     02(ER)     000000
                33    IHEUPA     02(ER)     000000
                34    IHEVQC     02(ER)     000000
```

Figure 15-14. Sample Output for LISTLOAD OUTPUT=MODLIST for a PDS (Part 1 of 4)

```
                              LISTING  OF  LOAD  MODULE  PL1LOAD                                      PAGE  0002

RECORD# 4     TYPE  01  -  CONTROL           CONTROL  SIZE  32                   CCW  06000000  40000780

               CESD#      LENGTH
                   1       04B8
                   2       0260
                   8       0008
                   9       0010
                  23       0008
                  24       0008
                  25       0008
                  26       0038
RECORD# 5                                                   TEXT
             000000      47F0F014  07D7D3F1  E3C3F0F2  000000D8      000004B8  90EBD00C  58B0F010  5800F00C
             000020      58F0B020  05EF05A0  4190D0B8  50DC0018      9200D062  9201D063  92C0D000  9202D063
             000040      F811D090  B132F810  D092B080  FA11D092      B130F821  D0A8D090  F821D0AB  D092D203
             000060      D0AEB134  F811D090  B13CF810  D092B080      FA11D092  B13AF821  D0B2D090  F821D0B5
             000080      D09241A0  A0600700  9203D063  4110B174      58F0B05C  05EF4110  B1144120  B18358F0
             0000A0      B05405EF  9203D063  58F0B058  05EF9204      D0635880  B070F821  D0908000  F821D093
             0000C0      8002FA20  D093B111  5870B06C  D2017000      D091D201  7002D094  9205D063  F821D090
             0000E0      7000F821  D0937002  FA20D093  B10F5860      B068D201  6000D091  D2016002  D0949206
             000100      D0634150  D0AE5050  D0944150  D0905050      D0989680  D0984110  D09458F0  B06405EF
             000120      5880B070  D2038000  D0909207  D063F811      D090B10C  F810D092  B080FA11  D092B10A
             000140      F9118000  D0904770  A0C8F911  8002D092      4780A0EE  9208D063  4110B168  58F0B05C
             000160      05EF4110  B14058F0  B05005EF  9208D063      58F0B058  05EF9208  D0639210  D0634180
             000180      D0A85080  D0984180  D0B25080  D09C4180      D0905080  D0A09680  D0A04110  D09858F0
             0001A0      B04005EF  D205D0B2  D0909211  D063D202      D090D0B2  F921D090  B0D19200  D0904780
             0001C0      A13E9280  D090D202  D091D0B5  F921D091      B0CF9200  D0914780  A1569280  D091D200
             0001E0      D094D090  D600D094  D0919180  D0944780      A19E9212  D0634110  B15C58F0  B05C05EF
             000200      4110B0A0  4120B183  58F0B054  05EF4110      D0B24120  B18758F0  B05405EF  9212D063
             000220      58F0B058  05EF9213  D0634110  B15058F0      B05C05EF  4110B084  4120B183  58F0B054
             000240      05EF9213  D06358F0  B05805EF  9214D063      58F0B030  05EF47F0  47F0F00C  03C1E7F1
             000260      000000D0  90EBD00C  18AF41E0  A0285830      B0381B22  50203050  58F0B02C  47F0F062
             000280      9201D084  58E01000  50E0D088  4580A03A      07FA05A0  4190D0B0  50DC001C  9200D062
             0002A0      9209D063  41A0A088  07F80700  47F0F00C      03C1C3F1  00000258  90EBD00C  58A0F008
             0002C0      45E0A016  9202D084  D207D0A0  10009200      D0A458E0  100850E0  D0884580  A03A47F0
             0002E0      A0000700  47F0F00C  03C1C3F2  00000258      90EBD00C  58A0F008  45E0A016  9203D084
             000300      D207D0A8  10009200  D0AC58E0  100850E0      D0884580  A03A47F0  A0860700  920BD063
             000320      920CD063  5880D0A0  F821D090  80005870      D0A4FA21  D0907000  F821D093  8002FA21
             000340      D0937002  9502D084  4780A062  9503D084      4780A076  5860D088  F872D098  D0904FE0
             000360      D09810FE  54E0B078  90EFD098  964ED098      2B006A00  D0987000  600047F0  A0805880
             000380      D088D201  8000D091  D2018002  D09447F0      A0805880  D088D205  8000D090  58F0B060
             0003A0      05EF920D  D063920E  D0635880  D0A8F822      D0908000  5870D0AC  FB22D090  7000F822
             0003C0      D0938003  FB22D093  70039502  D0844780      A0E89503  D0844780  A0FC5860  D088F872
             0003E0      D098D090  4FE0D098  10FE54E0  B07890EF      D098964E  D0982B00  6A00D098  70006000
             000400      47F0A106  5880D088  D2018000  D091D201      8002D094  47F0A106  5880D088  D2058000
             000420      D09058F0  B06005EF  920FD063  58F0B92C      05EFF014  9180D001  4780F03C  5820D050
             000440      12224770  F03C59DC  00104770  F03C58D0      D00450DC  00109180  D0004710  F03258D0
             000460      D00447F0  F0225020  D00898EB  D00C07FE      58F0B030  07FF584C  00001244  47B0F056
             000480      587C0014  D2033050  70504140  4001504C      00005040  30549200  304C5030  D00818D3
             0004A0      583C0010  5030D004  50DC0010  5020D008      5020D060  07FE1C44  00001000  000014B8
             0004C0      000024B8  000034B8  000044B8  000054B8      000064B8  000074B8  00000000  00000000
             0004E0      00000434  00000434  00000000  89300008      00000648  41660001  000002E4  000002AC
```

Figure 15-14. Sample Output for LISTLOAD OUTPUT=MODLIST for a PDS (Part 2 of 4)

```
                                          LISTING  OF  LOAD  MODULE  PL1LOAD                                  PAGE  0003
        000500     00000258  00000000  00000000  00000000     00000000  00000000  00000000  00000000
        000520     00000730  00000738  00000740  00000748     80000000  00000001  0C020000  00000544
        000540     00140014  40D7D3F1  E3C3F0F2  6060C3D6     D4D7D3C5  E3C5C440  0000Q560  00270027
        000560     40C5D9D9  6D9D6BC5  E7D7C5C3  E3C5C440     C1C440C9  E240F4F0  4EF2F0C9  40C2E4E3
        000580     40C1C440  C9E24002  0C040C00  00000594     002C002C  40C5D9D9  D6D96BC5  E7D7C5C3
        0005A0     E3C5C440  C140C9E2  40F1F84E  F4F1C940     C2E4E340  C140C9E2  40D9C5C1  D3D3E840
        0005C0     000C041C  018C0C2C  0C1C0000  000005D4     00120012  40D7D3F1  E3C3F0F2  6060C5D5
        0005E0     E3C5D9C5  C440000C  040C050C  000C006C     000C020C  010C001C  0000058C  0000063B
        000600     00000740  80000638  00000748  00000242     80000534  00000748  0000021C  80000534
        000620     00000748  0000016C  80000534  00000748     000000A4  80000534  8903802C  8A060089
        000640     04800620  41C90008  C08000D0  1C021AC1     95043008  47808200  D2AFC000  40009680
        000660     900647F0  8206D2AF  4000C000  1BFF50FD     00101817  41000038  0A0A98EC  D00C07FE
        000680     00033BC8  00480A0A  05804860  B08050E7     00309180  90064780  80189205  701047F0
        0006A0     801C9206  70104150  A05818C6  41D00020     1CCC1AD5  50D70014  184D9505  70104770
        0006C0     804048D0  900447F0  80581B22  8D200008     41100001  19128C20  00084780  809648D7
        0006E0     00224820  B07A4BD0  B0864740  807A1BCC     4810B07E  1DC11AD2  89D00008  41DCD001
        000700     47F0808A  4AD0B086  4AD0B084  06208920     00081AD2  410D0000  00000000  47F0809E
        000720     58F0F008  07FF0000  00000000  50070034     003C004C  001058F0  003C004C  58070034
        000740     003C004C  D2071024  00201002  00000000     00000004  00000000  00000000  00000000
        000760     07E2E8E2  D7D9C9D5  E3000000  00000000     00000000  00000000  00000000  00000000
                                                                   RLD SIZE 236
RECORD# 6      TYPE 02 - RLD
               R-PTR  P-PTR     FL  ADDR     FL  ADDR     FL  ADDR     FL  ADDR     FL  ADDR     FL  ADDR
                   2      1     0C 000010
                  14      1     24 00002E
                  15      1     24 00029A
                   1      1     0D 0002B4    0C 0002EC
                  12      1     25 000448    24 000454
                   3      1     24 000478
                  13      1     24 000482
                   3      1     24 000490
                  12      1     25 0004A2    24 0004AA
                   2      2     0D 0004BC    0D 0004C0    0D 0004C4    0D 0004C8    0D 0004CC    0D 0004D0
                                0C 0004D4
                   4      2     8C 0004D8
                   5      2     8C 0004DC
                   1      2     0D 0004E0    0C 0004E4
                   2      2     0C 0004F0
                   1      2     0D 0004F8    0D 0004FC    0D 000500    0C 000504
                  16      2     9C 000508
                  17      2     9C 00050C
                  18      2     9C 000510
                  19      2     9C 000514
                  20      2     9C 0004E8
                  21      2     9C 000518
                  22      2     9C 00051C
                  23      2     0C 000520
```

Figure 15-14. Sample Output for LISTLOAD OUTPUT=MODLIST for a PDS (Part 3 of 4)

```
                                LISTING  OF  LOAD  MODULE  PL1LOAD                                    PAGE  0004

RECORD# 7     TYPE  0E  -  RLD                                   RLD SIZE 236
              R-PTR  P-PTR    FL  ADDR    FL  ADDR    FL  ADDR    FL  ADDR    FL  ADDR    FL  ADDR
                24     2      0C 000524
                25     2      0C 000528
                26     2      0C 00052C
                 2     2      09 00053D   09 000559   09 00058D   09 0005CD   0D 0005F8   0C 0005FC
                25     2      0C 000600
                 2     2      08 000605
                26     2      0C 000608
                 1     2      0C 00060C
                 2     2      08 000611
                26     2      0C 000614
                 1     2      0C 000618
                 2     2      08 00061D
                26     2      0C 000620
                 1     2      0C 000624
                 2     2      08 000629
                26     2      0C 00062C
                 1     2      0C 000630
                 2     2      08 000635
                 1     8      0C 000718
                10     9      8C 000728
                27    26      24 000748

******END OF LOAD MODULE LISTING
```

Figure 15-14. Sample Output for LISTLOAD OUTPUT=MODLIST for a PDS (Part 4 of 4)

```
                                          LISTING OF PROGRAM OBJECT TESTPR                              PAGE      1

THIS PROGRAM OBJECT WAS ORIGINALLY PRODUCED BY 5695DF108 AT LEVEL 01.00 ON 09/16/92 AT 09:42:47
------------------------------------------------------------------------------------------------------------------------
           CONTROL SECTION:  A
AMODE: 24                 ALIGNMENT: DOUBLE WORD       LENGTH:         24 (DEC)        MODULE OFFSET:          0 (DEC)
RMODE: 24                 USABILITY: UNSPECIFIED       LENGTH:         18 (HEX)        MODULE OFFSET:          0 (HEX)
                          STORAGE:   ANY               OVERLAY SEGMENT:       0        OVERLAY REGION:       0
===== IDRL =====
          TRANSLATOR   VER   MOD     DATE
          566896201     02    01    09/16/92
===== ESDs =====
ALPHA(PR)
                ALIGNMENT: FULL WORD        LENGTH: 00000004
===== RLDs =====
SEC.OFF   MOD.OFF   TYPE     BDY    STATUS       REFERENCED SYMBOL
00000004  00000004   CPR     NONE    RES         $CUMULATIVE PSEUDO REGISTER LENGTH
00000008  00000008    PR     NONE    RES      (+)ALPHA
00000010  00000010   CPR     NONE    RES         $CUMULATIVE PSEUDO REGISTER LENGTH
00000014  00000014   CPR     NONE    RES         $CUMULATIVE PSEUDO REGISTER LENGTH
===== TEXT =====
00000000  C1C1C1C1 00000014 00000000 00000000    00000014 00000014
                                          LISTING OF PROGRAM OBJECT TESTPR                              PAGE      2

                PSEUDO REGISTER
                 VECTOR LOC   LENGTH    NAME
                        0        4    ALPHA
                        0       10    BETA
LENGTH OF PSEUDO REGISTERS       14
**   END OF PROGRAM OBJECT LISTING
```

Figure 15-15. Sample Output for LISTLOAD OUTPUT=MODLIST for a PDSE (Program Object Version 1)

```
                                NUMERICAL MAP AND CROSS-REFERENCE LIST OF PROGRAM OBJECT TESTPR           PAGE      1

      LMOD LOC   SECT LOC    LENGTH  TYPE           NAME
        0                      18   SD             A
------------------------------------------------------------------------------------------------------------------------
         LMOD LOC   SECT LOC
ADCON AT       4         4           IN SECTION  A
             $CUM PSEUDO REGISTER      REFERS TO  $CUMULATIVE PSEUDO REGISTER LENGTH
ADCON AT       8         8           IN SECTION  A
             $PSEUDO REGISTER          REFERS TO  ALPHA
ADCON AT      10        10           IN SECTION  A
             $CUM PSEUDO REGISTER      REFERS TO  $CUMULATIVE PSEUDO REGISTER LENGTH
ADCON AT      14        14           IN SECTION  A
             $CUM PSEUDO REGISTER      REFERS TO  $CUMULATIVE PSEUDO REGISTER LENGTH
LENGTH OF PROGRAM OBJECT       18
                                NUMERICAL MAP AND CROSS-REFERENCE LIST OF PROGRAM OBJECT TESTPR           PAGE      2

                PSEUDO REGISTER
                 VECTOR LOC   LENGTH    NAME
                         0       10    BETA
                         0        4    ALPHA
LENGTH OF PSEUDO REGISTERS        14
                                           ALPHABETICAL MAP OF PROGRAM OBJECT TESTPR                      PAGE      3

ENTRY NAME  LMOD LOC  SECT LEN/LOC            SECTION NAME OR ENTRY TYPE
A
                  0        18                (SD)
------------------------------------------------------------------------------------------------------------------------
                                   ALPHABETICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT TESTPR             PAGE      4

      LMOD LOC  SECT LOC            SYMBOL
     $CUM PSEUDO REGISTER          $CUMULATIVE PSEUDO REGISTER LENGTH
            4         4                REFERENCED IN  A
     $CUM PSEUDO REGISTER          $CUMULATIVE PSEUDO REGISTER LENGTH
           10        10                REFERENCED IN  A
     $CUM PSEUDO REGISTER          $CUMULATIVE PSEUDO REGISTER LENGTH
           14        14                REFERENCED IN  A
     $PSEUDO REGISTER              ALPHA
            8         8                REFERENCED IN  A
                                   ALPHABETICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT TESTPR             PAGE      5

                PSEUDO REGISTER
                 VECTOR LOC   LENGTH    NAME
                         0        4    ALPHA
                         0       10    BETA
**   END OF MAP AND CROSS-REFERENCE LISTING
```

Figure 15-16. Sample Output for LISTLOAD OUTPUT=XREF for a PDSE (Program Object Version 1)

```
   LISTLOAD  MEMBER=TESTLR5,OUTPUT=BOTH
                               ***** M O D U L E   S U M M A R Y *****
    MEMBER NAME:  TESTLR5                                                                      MAIN ENTRY POINT:    00000028
    LIBRARY:      SYSLIB                                                                       AMODE OF MAIN ENTRY POINT: 24
        NO ALIASES **
------------------------------------------------------------------------------------------------------------------------
                       ****         ATTRIBUTES OF MODULE          ****
                 **   BIT  STATUS        BIT  STATUS        BIT  STATUS        BIT  STATUS   **
                       0  NOT-RENT        1  NOT-REUS        2  NOT-OVLY        3  NOT-TEST
                       4  NOT-OL          5  BLOCK           6  EXEC            7  MULTI-RCD
                       8  NOT-DC          9  ZERO-ORG        10 RESERVED        11 RLD
                       12 NOT-EDIT        13 NO-SYMS         14 RESERVED        15 NOT-REFR
                       16 RESERVED        17 <16M            18 NOT-PL          19 NO-SSI
                       20 NOT-APF         21 PGM OBJ         22 RESERVED        23 RESERVED
                       24 NOT-ALTP        25 RESERVED        26 RESERVED        27 RMODE24
                       28 RESERVED        29 RESERVED        30 RESERVED        31 RESERVED
                       32 MIGRATE         33 NO-PRIME        34 NO-PACK         35 RESERVED
                       36 RESERVED        37 RESERVED        38 RESERVED        39 RESERVED
------------------------------------------------------------------------------------------------------------------------
-                                    MODULE SSI:  NONE
                                     APFCODE:     00000000
                                     RMODE:       24
                                     PO FORMAT:   1
                                     XPLINK:      NO
                  *****PROGRAM OBJECT PROCESSED BY BINDER
***THE FOLLOWING ARE THE UNFORMATTED PDSE DIRECTORY ENTRY SECTIONS (PMAR AND PMARL)
PMAR  001E0107 02C80400 00000000 00300000   00280000 00280000 00000000 0000
PMARL 00320080 00000000 00010000 002C0000   01900000 00000000 01C00000 003C0000
      013C0000 00200000 011C0000 00140000   0178
                                           LISTING OF PROGRAM OBJECT TESTLR5                               PAGE      1

THIS PROGRAM OBJECT WAS ORIGINALLY PRODUCED BY 5695DF108 AT LEVEL 02.10 ON 03/13/2001 AT 15:02:16
------------------------------------------------------------------------------------------------------------------------
           MODULE SECTION:   $SUMMARY
USABILITY: UNSPECIFIED    OVERLAY SEGMENT:      0     OVERLAY REGION:      0
===== ESDs =====
B_PRV(ED)
    CLASS:           B_PRV     LENGTH:          4 (HEX)      CLASS OFFSET:           0 (HEX)      FORMAT:  F(0001)
    NAME SPACE:          2     ALIGNMENT:   DOUBLE WORD      BIND METHOD:              MERGE      RMODE:       UNS
    TEXT                       NOLOAD                        FILL:                    UNSPEC
Q1(PD)
    CLASS:           B_PRV     LENGTH:          4 (HEX)      CLASS OFFSET:           0 (HEX)
    NAME SPACE:          2     ALIGNMENT:     FULL WORD      PRIORITY:               0 (HEX)      SCOPE:    UNSPEC
    ATTRIBUTES:     WEAK
------------------------------------------------------------------------------------------------------------------------
           CONTROL SECTION:  A
USABILITY: UNSPECIFIED    OVERLAY SEGMENT:      0     OVERLAY REGION:      0
===== IDRL =====
          TRANSLATOR   VER   MOD     DATE
          566896201     02    01    03/13/2001
===== ESDs =====
B_TEXT(ED)
    CLASS:          B_TEXT     LENGTH:         18 (HEX)      CLASS OFFSET:           0 (HEX)      FORMAT:  F(0001)
    NAME SPACE:          1     ALIGNMENT:   DOUBLE WORD      BIND METHOD:           CATENATE      RMODE:        24
    TEXT                       LOAD                          FILL:                    UNSPEC
A(LD)
    CLASS:          B_TEXT     TEXT TYPE:        UNSPEC      CLASS OFFSET:           0 (HEX)
    NAME SPACE:          1     SCOPE:            MODULE      ELEMENT OFFSET:         0 (HEX)      AMODE:        24
    ATTRIBUTES:     GENERATED,STRONG
ENTA(LD)
    CLASS:          B_TEXT     TEXT TYPE:        UNSPEC      CLASS OFFSET:          14 (HEX)
    NAME SPACE:          1     SCOPE:            MODULE      ELEMENT OFFSET:        14 (HEX)      AMODE:        24
    ATTRIBUTES:     STRONG
ENTA(ER)
                               TEXT TYPE:        UNSPEC      CLASS OFFSET:          14 (HEX)
    TARGET SECTION: A                                        TARGET CLASS:            B_TEXT
    NAME SPACE:          1     SCOPE:           LIBRARY      ELEMENT OFFSET:        14 (HEX)
    RESOLVED                   AUTOCALL
    ATTRIBUTES:     STRONG
```

Figure 15-17. Sample Output for LISTLOAD OUTPUT=BOTH for a .... (Part 1 of 4)

```
ENTB(ER)
                                 TEXT TYPE:         UNSPEC      CLASS OFFSET:          28 (HEX)
    TARGET SECTION: B                                           TARGET CLASS:           B_TEXT
    NAME SPACE:           1      SCOPE:            LIBRARY      ELEMENT OFFSET:        10 (HEX)
    RESOLVED                     AUTOCALL
    ATTRIBUTES:     STRONG
B_PRV(ED)
    CLASS:            B_PRV      LENGTH:            0 (HEX)     CLASS OFFSET:           0 (HEX)       FORMAT:  F(0001)
                                          LISTING OF PROGRAM OBJECT TESTLR5                                    PAGE      2
           CONTROL SECTION:  A
===== ESDs =====
    NAME SPACE:           2      ALIGNMENT:    DOUBLE WORD      BIND METHOD:             MERGE        RMODE:       UNS
    TEXT                         NOLOAD                         FILL:                   UNSPEC
Q1(PR)
    CLASS:            B_PRV      LENGTH:            4 (HEX)     CLASS OFFSET:           0 (HEX)
    NAME SPACE:           2      ALIGNMENT:      FULL WORD      PRIORITY:               0 (HEX)       SCOPE:   UNSPEC
    ATTRIBUTES:     WEAK
===== RLDs =====            CLASS:           B_TEXT
ELEM.OFF CLS.OFF  TYPE STATUS LENG HOBCHG NSPACE      TARGET NAME      PARTRES              XATTR NAME       XATTR OFF
00000006 00000006   BR   RES  0004   NO      1     (+)ENTA
0000000C 0000000C C-OF   RES  0004   NO      2     (+)Q1
00000010 00000010   BR   RES  0004   NO      1     (+)ENTB
===== TEXT =====            CLASS:           B_TEXT
00000000  07FEC1C1 C1C10000 00140000 00000000 00000028 C5D5E3C1                      ..AAAA..............ENTA........*
------------------------------------------------------------------------------------------------------------------------
           CONTROL SECTION:  B
USABILITY: UNSPECIFIED    OVERLAY SEGMENT:       0     OVERLAY REGION:       0
===== IDRL =====
          TRANSLATOR   VER   MOD     DATE
          566896201     02    01    03/13/2001
===== ESDs =====
B_TEXT(ED)
    CLASS:           B_TEXT      LENGTH:           14 (HEX)     CLASS OFFSET:          18 (HEX)       FORMAT:  F(0001)
    NAME SPACE:           1      ALIGNMENT:    DOUBLE WORD      BIND METHOD:          CATENATE        RMODE:        24
    TEXT                         LOAD                           FILL:                   UNSPEC
B(LD)
    CLASS:           B_TEXT      TEXT TYPE:         UNSPEC      CLASS OFFSET:          18 (HEX)
    NAME SPACE:           1      SCOPE:             MODULE      ELEMENT OFFSET:         0 (HEX)       AMODE:        24
    ATTRIBUTES:     GENERATED,STRONG
ENTB(LD)
    CLASS:           B_TEXT      TEXT TYPE:         UNSPEC      CLASS OFFSET:          28 (HEX)
    NAME SPACE:           1      SCOPE:             MODULE      ELEMENT OFFSET:        10 (HEX)       AMODE:        24
    ATTRIBUTES:     STRONG
ENTB(ER)
                                 TEXT TYPE:         UNSPEC      CLASS OFFSET:          28 (HEX)
    TARGET SECTION: B                                           TARGET CLASS:           B_TEXT
    NAME SPACE:           1      SCOPE:            LIBRARY      ELEMENT OFFSET:        10 (HEX)
    RESOLVED                     AUTOCALL
   ATTRIBUTES:     STRONG
UNRES(ER)
                                 TEXT TYPE:         UNSPEC      CLASS OFFSET:           0 (HEX)
    TARGET SECTION:                                             TARGET CLASS:           B_TEXT
    NAME SPACE:           1      SCOPE:            LIBRARY      ELEMENT OFFSET:         0 (HEX)
                                          LISTING OF PROGRAM OBJECT TESTLR5                                    PAGE      3
```

Figure 15-17. Sample Output for LISTLOAD OUTPUT=BOTH for a .... (Part 2 of 4)

```
           CONTROL SECTION:  B
===== ESDs =====
    UNRESOLVED                  AUTOCALL
    ATTRIBUTES:     STRONG
===== RLDs =====            CLASS:          B_TEXT
ELEM.OFF CLS.OFF  TYPE STATUS LENG HOBCHG NSPACE      TARGET NAME      PARTRES               XATTR NAME       XATTR OFF
00000006 0000001E   BR   RES  0004   NO      1     (+)ENTB
0000000C 00000024   BR UNRES  0004   NO      1     (+)UNRES
===== TEXT =====            CLASS:          B_TEXT
00000018  07FEC2C2 C2C20000 00280000 00000000 C5D5E3C2                           ..BBBB..........ENTB............*
**   END OF PROGRAM OBJECT LISTING
                             ** NUMERICAL MAP OF PROGRAM OBJECT TESTLR5          **                   PAGE      1

------------------------------------------------------------------------------------------------------------------------
RESIDENT CLASS:           B_TEXT
      CLAS LOC   ELEM LOC     LENGTH  TYPE    ALIGNMENT             NAME
        0                         18   ED     DOUBLE WORD           A
            14         14              LD                             ENTA
       18                         14   ED     DOUBLE WORD           B
            28         10              LD                             ENTB
      CLASS LENGTH                30
------------------------------------------------------------------------------------------------------------------------
                             ** NUMERICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT TESTLR5         **       PAGE      2

         CLAS LOC   ELEM LOC         TARGET CLASS
ADCON AT       6         6                            IN SECTION  A
              14        14             B_TEXT           REFERS TO  ENTA
               0         0                                IN SECTION  A
ADCON AT       C         C                            IN SECTION  A
             $CLASS_OFFSET              B_PRV           REFERS TO  Q1
ADCON AT      10        10                            IN SECTION  A
              28        10             B_TEXT           REFERS TO  ENTB
              18         0                                IN SECTION  B
ADCON AT      1E         6                            IN SECTION  B
              28        10             B_TEXT           REFERS TO  ENTB
              18         0                                IN SECTION  B
ADCON AT      24         C                            IN SECTION  B
               *         *              ****           REFERS TO  $UNRESOLVED
------------------------------------------------------------------------------------------------------------------------
RESIDENT CLASS:            B_PRV
      CLAS LOC   ELEM LOC     LENGTH  TYPE    ALIGNMENT             NAME
        0                          4   ED     DOUBLE WORD           $PRIV000003
             0                     4   PD     FULL WORD               Q1
        0                          0   ED     DOUBLE WORD           A
      CLASS LENGTH                 0
------------------------------------------------------------------------------------------------------------------------
                             ** NUMERICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT TESTLR5         **       PAGE      3

     ****  NO ADCONS IN THIS CLASS  ****
LENGTH OF PROGRAM OBJECT       30
------------------------------------------------------------------------------------------------------------------------
                             ** ALPHABETICAL MAP OF PROGRAM OBJECT TESTLR5          **                    PAGE      4

ENTRY NAME  CLAS LOC  ELEM LEN/LOC            CLASS NAME      SECTION NAME OR ENTRY TYPE
$PRIV000003
                  0          4                     B_PRV      (ED)
A
                  0         18                    B_TEXT      (ED)
A
                  0          0                     B_PRV      (ED)
B
                 18         14                    B_TEXT      (ED)
ENTA
                 14              14               B_TEXT      A
ENTB
                 28              10               B_TEXT      B
Q1
                  0          4                     B_PRV      (PD)
```

Figure 15-17. Sample Output for LISTLOAD OUTPUT=BOTH for a .... (Part 3 of 4)

```
--------------------------------------------------------------------------------------------------------------------------
                             ** ALPHABETICAL CROSS-REFERENCE LIST OF PROGRAM OBJECT TESTLR5         **    PAGE      5

      CLAS LOC  ELEM LOC           TARGET CLASS       SYMBOL
        *             *                    ****       $UNRESOLVED
       14            14              B_TEXT           ENTA
            0         0                                 IN SECTION  A
            6         6                                   REFERENCED IN  A
       28            10              B_TEXT           ENTB
           18         0                                 IN SECTION  B
           10        10                                   REFERENCED IN  A
       28            10              B_TEXT           ENTB
           18         0                                 IN SECTION  B
           1E         6                                   REFERENCED IN  B
      $CLASS_OFFSET                   B_PRV           Q1
            C         C                                   REFERENCED IN  A
LENGTH OF PROGRAM OBJECT       30
**   END OF MAP AND CROSS-REFERENCE LISTING
```

Figure 15-17. Sample Output for LISTLOAD OUTPUT=BOTH for a .... (Part 4 of 4)

Chapter 16. SPZAP

A laser that allows you to perform surgery on the system. . . . and you don't need a PhD!

SPZAP is a service aid program that operates in problem state. SPZAP allows you to dynamically update and maintain programs and data sets. SPZAP can be used to apply fixes to modules or programs that need to be at current levels of the operating system.

The functions of SPZAP provide many capabilities, including:

- Using the inspect and modify functions of SPZAP, you can fix programming errors that require only the replacement of instructions in a load module member of a PDS or a program object member of a PDSE without recompiling the program.
- Using the modify function of SPZAP, you can set traps in a program by inserting incorrect instructions. The incorrect instructions will force abnormal ending; the dump of storage provided as a result of the abnormal ending is a valuable diagnostic tool, because it shows the contents of storage at a predictable point during processing.
- Using SPZAP to replace data directly on a direct access device, you can reconstruct VTOCs or data records that may have been destroyed as the result of an I/O error or a programming error.
- On the advice of the IBM Support Center, start tracing in system components that do not use component trace. The IBM Support Center will tell you how to use the SPZAP service aid to start traces in these components.
- Update the system status index (SSI) in the directory entry for any load module in a PDS or program object in a PDSE. Update the information record (IDR) in any load module in a PDS or program object in a PDSE.

Major Topics

This chapter describes each of the following topics:

- "Planning for SPZAP"
- "Inspecting and Modifying Data" on page 16-2
- "Updating the System Status Index (SSI)" on page 16-13
- "Running SPZAP" on page 16-14
- "Reading SPZAP Output" on page 16-29

Planning for SPZAP

SPZAP is an application that provides editing capabilities for data on a direct access storage device (DASD). Protect against SPZAP (and other applications that can update data sets) being used to damage data through use of the installation's security protection scheme:

- In *z/OS DFSMS: Using Data Sets*, see the chapter, "Protecting Data Sets" for information pertaining to protecting data sets.
- In *z/OS DFSMSdfp Advanced Services*, see the chapter, "Protecting the VTOC and VTOC Index" for information pertaining to protecting VTOCs.

Installations using RACF should employ a combination of GDASDVOL and DASDVOL resource profiles to establish this protection. See *z/OS Security Server RACF Security Administrator's Guide* for more information regarding these profiles.

IBM recognizes the particular sensitivity of the VTOC. For a VTOC, the console operator must respond to message AMA117D before SPZAP will process an update request. This authorization must be supplied in addition to authorization through use of the installation's security protection scheme.

Inspecting and Modifying Data

The inspection function is controlled by the VERIFY statement. VERIFY allows you to check the contents of a specific location in a load module member of a PDS, a program object member of a PDSE, a specific physical record of a direct access data set, or a record of a member of a data PDSE before you replace the contents. If the contents at the specified location do not agree with the contents as specified in the VERIFY statement, subsequent REP operations are not performed.

The SPZAP modification function is controlled by the REP (replace) control statement. The REP control statement allows you to replace instructions or data at a specific location in a load module member of a PDS, a program object in a PDSE, a physical record in a direct access data set or a record of a member of a data PDSE.

To avoid possible errors in replacing data, you should always precede any REP operation with a VERIFY operation.

This topic describes the following:
- "Inspecting and Modifying a Load Module"
- "Inspecting and Modifying a Data Record" on page 16-10

Inspecting and Modifying a Load Module

To inspect or modify data in a load module member of a PDS or program object member of a PDSE, you need a NAME statement to supply SPZAP the name of the appropriate member. The load module or program object must be a member of the PDS or PDSE, respectively, identified by the SYSLIB DD statement included in the JCL.

If the load module member of a PDS or program object member of a PDSE contains more than one control section (CSECT), you must also supply SPZAP with the name of the CSECT that is to be inspected or modified. If no CSECT name is given in the NAME statement, SPZAP assumes that the control section to be processed is the first one encountered in searching the load module.

Whenever SPZAP updates a CSECT in a load module member of a PDS or program object member of a PDSE in response to your NAME and REP control statements, it also puts descriptive maintenance data in a CSECT identification record (IDR) associated with the load module or program object. This function will be performed automatically after all REP statements associated with the NAME statement have been processed; any optional user data that has to be placed in the IDR will come from the IDRDATA statement. See "SPZAP Control Statements" on page 16-17 for an explanation of the IDRDATA statement.

Example: Inspecting and Modifying a Single CSECT Load Module

This example shows how to inspect and modify a load module containing a single CSECT.

```
//ZAPCSECT     JOB        MSGLEVEL=(1,1)
//STEP         EXEC       PGM=AMASPZAP
//SYSPRINT     DD         SYSOUT=A
//SYSLIB       DD         DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSIN        DD         *
 NAME       IEEVLNKT
 VERIFY     0018      C9C8,D2D9,D1C2,C7D5
 REP        0018      E5C6,D3D6,E6F0,4040
 SETSSI     01211234
 IDRDATA    71144
 DUMP       IEEVLNKT
/*
```

SYSLIB DD Statement: Defines the system library SYS1.LINKLIB containing the module IEEVLNKT that SPZAP is to process.

NAME Control Statement: Instructs SPZAP that the operations defined by the control statements that follow are to be performed on the module IEEVLNKT.

VERIFY Control Statement: Requests that SPZAP check the hexadecimal data at offset X'0018' in the module IEEVLNKT to make sure that it is the same as the hexadecimal data specified in this statement. If the data is the same, SPZAP continues processing the subsequent statements sequentially. If the data is not identical, SPZAP will not perform the REP and SETSSI operations requested for the module. It will, however, perform the requested DUMP operation before discontinuing the processing. It will also dump a hexadecimal image of the module IEEVLNKT to the SYSPRINT data set.

REP Control Statement: Causes SPZAP to replace the data at offset X'0018' in module IEEVLNKT with the data given in this control statement, provided the VERIFY statement was successful.

SETSSI Control Statement: Instructs SPZAP to replace the system status information in the directory entry for module IEEVLNKT with the SSI data given in the statement, if the VERIFY statement was successful. The new SSI is to contain:
- A change level of 01
- A flag byte of 21
- A serial number of 1234

IDRDATA Control Statement: Causes SPZAP to update the IDR in module IEEVLNKT with the data 71144, if the REP operation is successful.

DUMP Control Statement: Requests that a hexadecimal image of module IEEVLNKT be dumped to the SYSPRINT data set. Since the DUMP statement follows the REP statement, the image will reflect the changes made by SPZAP if the VERIFY operation was successful.

Example: Modifying a CSECT in a Load Module

This example shows how to apply an IBM-supplied PTF in the form of an SPZAP fix, rather than a module replacement PTF.

```
//PTF40228   JOB      MSGLEVEL=(1,1)
//STEP       EXEC     PGM=AMASPZAP
//SYSPRINT   DD       SYSOUT=A
//SYSLIB     DD       DSNAME=SYS1.NUCLEUS,DISP=OLD
//SYSIN      DD       *
 NAME      IEANUC01  IEWFETCH
 IDRDATA   LOCFIX01
 VERIFY    01F0 47F0C018
 VERIFY    0210 5830C8F4
 REP       01F0 4780C072
 REP       0210 4130C8F4
 SETSSI    02114228
 DUMPT     IEANUC01  IEWFETCH
/*
```

SYSLIB DD Statement: Defines the library (SYS1.NUCLEUS) that contains input module IEANUC01.

SYSIN DD Statement: Defines the input stream.

NAME Control Statement: Instructs SPZAP that the operations defined by the control statements that immediately follow this statement are to be performed on the CSECT IEWFETCH contained in the load module IEANUC01.

IDRDATA Control Statement: Causes SPZAP to update the IDR in module IEANUC01 for CSECT IEWFETCH with the date LOCFIX01, if either of the REP operations is successful.

VERIFY Control Statements: Requests that SPZAP compare the contents of the locations X'01F0' and X'0210' in the control section IEWFETCH with the data given in the VERIFY control statements. If the comparisons are equal, SPZAP continues processing subsequent control statements sequentially. However, if the data at the locations does not compare identically to the data given in the VERIFY control statements, SPZAP dumps a hexadecimal image of CSECT IEWFETCH to the SYSPRINT data set; the subsequent REP and SETSSI statements are ignored. The DUMPT function specified will be performed before SPZAP ends processing.

REP Control Statements: Causes SPZAP to replace the data at offsets X'01F0' and X'0210' from the start of CSECT IEWFETCH with the hexadecimal data specified on the corresponding REP statements.

SETSSI Control Statement: Causes SPZAP to replace the system status information in the directory for module IEANUC01 with the SSI data given in the SETSSI statement after the replacement operations have been effected. The new SSI will contain a change level of 02, a flag byte of 11, and a serial number of 4228.

DUMPT Control Statement: Causes SPZAP to produce a translated dump for CSECT IEWFETCH of load module IEANUC01.

Example: Inspecting and Modifying Two CSECTs (Part 1)

Use this JCL to inspect and modify two CSECTs in the same load module.

```
//CHANGIT    JOB        MSGLEVEL=(1,1)
//STEP       EXEC       PGM=AMASPZAP
//SYSPRINT   DD         SYSOUT=A
//SYSLIB     DD         DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSIN      DD         *
 NAME      IEFX5000  IEFQMSSS
 VERIFY    0284 4780,C096
 REP       0284 4770,C096
 IDRDATA   PTF01483
 SETSSI    01212448
 DUMPT     IEFX5000  IEFQMSSS
 NAME      IEFX5000  IEFQMRAW
 VERIFY    0154 4780,C042
 REP       0154 4770,C042
 IDRDATA   PTF01483
 SETSSI    01212448
 DUMPT     IEFX5000  IEFQMRAW
/*
```

SYSLIB DD Statement: Defines the system library SYS1.LINKLIB containing the load module IEFX5000 that is to be changed by SPZAP.

Example: Explanation of First Control Statements (Part 2)

NAME Control Statement #1: Instructs SPZAP that the operations requested via the control statements immediately following it are to be performed on CSECT IEFQMSSS in load module IEFX5000.

VERIFY Control Statement #1: Requests that SPZAP check the hexadecimal data at offset X'0284' in CSECT IEFQMSSS to make sure it is the same as the data specified in this control statement. If the data is identical, SPZAP continues processing the control statements. If the data is not identical, SPZAP does not perform the REP or SETSSI for CSECT IEFQMSSS, but it does perform the DUMPT operation. It also provides a hexadecimal dump of CSECT IEFQMSSS.

REP Control Statement #1: Causes SPZAP to replace the data at offset X'0284' in CSECT IEFQMSSS with the hexadecimal data given in this control statement.

IDRDATA Control Statement #1: Causes SPZAP to update the IDR in module IEFX5000 for CSECT IEFQMSSS with the data PTF01483, if the first REP operation is successful.

SETSSI Control Statement #1: Instructs SPZAP to replace the system status information in the directory entry for module IEFX5000 with the SSI data given. The new SSI will contain a change level of 01, a flag byte of 21, and a serial number of 2448.

DUMPT Control Statement #1: Provides a translated dump of CSECT IEFQMSSS.

Example: Explanation of Second Control Statements (Part 3)

NAME Control Statement #2: Indicates that the operations defined by the control statements that immediately follow this statement are to be performed on CSECT IEFQMRAW in the load module IEFX5000.

VERIFY Control Statement #2: Requests that SPZAP perform the VERIFY function at offset X'0154' from the start of CSECT IEFQMRAW. If the VERIFY operation is successful, SPZAP continues processing the subsequent control statements sequentially. If the VERIFY is rejected, however, SPZAP does not perform the following REP or SETSSI operations, but it does dump a hexadecimal image of CSECT IEFQMRAW to the SYSPRINT data set and performs the DUMPT operation as requested.

REP Control Statement #2: Causes SPZAP to replace the data at hexadecimal offset X'0154' from the start of CSECT IEFQMRAW with the hexadecimal data that is specified in this control statement.

IDRDATA Control Statement #2: Causes SPZAP to update the IDR in module IEFX5000 for CSECT IEFQMRAW with the data PTF01483, if the second REP operation is successful.

SETSSI Control Statement #2: Causes SPZAP to perform the same function as the previous SETSSI, but only if the second VERIFY is not rejected.

DUMPT Control Statement #2: Causes SPZAP to perform the DUMPT function on control section IEFQMRAW.

Example: Inspecting and Modifying a CSECT in HFS Prog. Obj.

Use this JCL to inspect and modify control section PRINTF in HFS program object LOADMOD1.

```
//ZAPHFS   EXEC PGM=AMASPZAP
//SYSPRINT DD  SYSOUT=A
//SYSLIB   DD  PATH='/sj/sjpl/binder/hfszap/',
// PATHDISP=(KEEP,KEEP),PATHOPTS=(ORDWR,ORDW)
//SYSIN    DD  *
  NAME     LOADMOD1  PRINTF
  VERIFY   0000 58F0C210
  REP      0000 68F0D210
  DUMP     LOADMOD1 PRINTF
/*
```

SYSLIB DD Statement: Defines the directory '/sj/sjpl/binder/hfszap/' containing the program object LOADMOD1 that SPZAP is to process.

SYSIN DD Statement: Defines the input stream.

NAME control statement: Instructs SPZAP that the operations defined by the control statements that follow are to be performed on the control section PRINTF of the program object LOADMOD1.

VERIFY control statement: Requests that SPZAP compare the contents of the location X'0000' in the control section PRINTF with the data given on the VERIFY control statement. If the comparisons are equal, SPZAP continues processing subsequent control statements sequentially. If the data does not compare, SPZAP dumps a hexadecimal image of CSECT PRINTF to the SYSPRINT data set; the subsequent REP control statement is ignored.

REP control statement: Causes SPZAP to replace the data at offset X'0000' from the start of the CSECT PRINTF with the hexadecimal data provided.

DUMP control statement: Requests that a hexadecimal image of program object LOADMOD1, control section PRINTF be dumped to the SYSPRINT data set. Because the dump statement follows the REP statement, the image will reflect the changes made by SPZAP if the VERIFY operation was successful.

Example: Using SPZAP to Modify a CSECT (Part 1)

Use this JCL to inspect and modify a CSECT within a program object module.

```
//UPDATE  JOB   MSGLEVEL=(1,1)
//ZAPSTEP EXEC  PGM=AMASPZAP
//SYSPRINT  DD  SYSOUT=A
//SYSLIB    DD  DSN=SYS1.LINKLIB,DISP=OLD
//SYSIN     DD  *
  NAME                                                          LONG#
                ALIASNAME       PDSPROCR
  VERIFY    000070   58E0,9118
  REP       000074   50E0,9434,9140,9058,47E0,C0A8,45E0,C476,94BF,9058,#
                     181D,58D0,D004,1FFF,43F0,A046,1F00,BF07,A047
  REP       00009A   1861,1870,1F55,0E64,98EC,D00C,07FE
```

SYSLIB DD statement: Defines the system library SYS1.LINKLIB containing a program object with an alias of LONGALIASNAME. (Note the continuation character (#) following LONG.) One CSECT in this program object is being changed.

SYSIN DD statement: Defines the input stream.

NAME control statement: This control statement contains a '#' in column 72 and is continued to a second control statement. The first 18 columns of the continued statement are blanks and are ignored. The string ALIASNAME on this continued statement is concatenated with the string LONG to form member name LONGALIASNAME. Note that this statement could have been contained in one record as NAME LONGALIASNAME PDSPROCR. Either way, the NAME statement indicates that SPZAP is to use the VERIFY and two REP statements to one CSECT PDSPROCR in the program object member whose alias is LONGALIASNAME.

Note: Leading blanks on the continued statement are ignored. No characters on the first card are skipped. Therefore, in order to split an operand, part on the first card and the rest on the second, it is important that the part of the operand on the first card extends to column 71. A blank in column 71 indicates that the non-blank string in the second card begins a new operand.

VERIFY control statement: Requests that SPZAP check the data at hexadecimal displacement X'000070' from the start of the data record defined in the CCHHR statement to make sure it is the same as the hexadecimal data specified in this control statement. If the data is the same, SPZAP continues processing the following control statements sequentially. If the data is not identical, SPZAP does not perform the REP function but does perform the ABSDUMPT operation; it also dumps a formatted hexadecimal image of the data record defined by the CCHHR statement to the SYSPRINT data set.

Example: Using SPZAP to Modify a CSECT (Part 2)

REP Control Statement #1: Causes SPZAP to replace the data at offset X'000074' in CSECT PDSPROCR with the hexadecimal data given in this control statement. Notice that this statement contains a non-blank (#) in column 72 indicating that it is continued to a second control statement.

REP Control Statement #2: Causes SPZAP to replace the data at offset X'00009A' in CSECT PDSPROCR with the hexadecimal data given in this control statement.

Accessing a Load Module

For a complete description of the control statements mentioned in the following discussion, see “SPZAP Control Statements” on page 16-17.

Once the CSECT has been found, the use of offset parameters in the VERIFY and REP statements allow SPZAP to locate the data that is to be verified and replaced. The offset parameters are specified in hexadecimal notation and define the displacement of the data relative to the beginning of the CSECT. For example, if a hexadecimal offset of X'40' is specified in a VERIFY statement, SPZAP will find the location that is 64 bytes beyond the beginning of the CSECT identified by the NAME statement, and begin verifying the data from that point.

Normally, the assembly listing address associated with the instruction to be inspected or modified can be used as the offset value in the VERIFY or REP statement. However, if a CSECT has been assembled with other CSECTs so that its origin is not at assembly location zero, then the locations in the assembly listing do not reflect the correct displacements of data in the CSECT. You must compute the proper displacements by subtracting the assembly listing address delimiting the start of the CSECT from the assembly listing address of the data to be referenced.

You can, however, use the BASE control statement to eliminate the need for such calculations and allow you to use the assembly listing locations. The BASE control statement should be included in the input to SPZAP immediately following the NAME statement that identifies the CSECT. The parameter in the BASE statement must be the assembly listing address (in hexadecimal) at which the CSECT begins. SPZAP then subtracts this value from the offset specified on any VERIFY or REP statement that follows the BASE statement, and uses the difference as the displacement of the data.

Figure 16-1 on page 16-10 is a sample assembly listing showing more than one control section. To refer to the second CSECT (IEFCVOL2), you could include in the input to SPZAP a BASE statement with a location of 0398. Then, to refer to the subsequent LOAD instruction (`L R2,CTJCTAD`) you could use an offset of X'039A' in the VERIFY or REP statements that follow in the SPZAP input stream.

```
      LISTING TITLE

 LOC   OBJECT CODE    ADDR1 ADDR2 STMT   SOURCE STATEMENT

000000                              1 IEFCVOL1 CSECT                 10000017

                                   .
                                   .
                                   .

000384 00000000                   378 VCNQMSSS DC   V(IEFQMSSS)      55800017
                                  379 *                              56000017
000388 00000000                   380 VCMSG15  DC   V(IEFVMG15)      56100017
00038C D200 1001 8000 00000 00000  381 MVCMSG   MVC  0(1,R1),0(R8)    56200017
                                  382 *                              56300017
000392 D200 1001 1000 00001 00000  383 MVCBLNKS MVC  1(1,R1),0(R1)    56400017
                                  384 *                              56500017



000398                            386          CSECT                 56600017
000398 0590                       387          BALR  R9.0            56700017
00039A                            388          USING *,R9            56800017
00039A 5820 C010           00010  389          L     R2,LCTJCTAD     56900017

                                   .
                                   .
                                   .
```

Figure 16-1. Sample Assembly Listing Showing Multiple Control Sections

Inspecting and Modifying a Data Record

You will inspect and modify a data record differently depending on whether the data record is in a PDS or a PDSE.

Record in a PDS

To inspect or modify a specific data record in a PDS you must use a CCHHR control statement to specify its direct access address. This CCHHR address must be within the limits of the direct access data set defined in the SYSLIB DD control statement.

If you request a REP operation for a record identified by a CCHHR control statement, SPZAP issues message AMA112I to provide a record of your request.

Record in a PDSE

To inspect or modify a specific data record in a PDSE data library, you must use the RECORD control statement preceded by a NAME control statement to specify its direct access address. This combination of RECORD and NAME serves as a pointer to a specific location in a PDSE data library member. The CCHHR control statement does not apply to a PDSE.

To determine the relative record number for a specific record, invoke SPZAP, specifying:

```
NAME membernam
ABSDUMP(T)  1 99999999
```

The results show a display of all records in the member, record length, relative record number, and other pertinent information.

Example: Inspecting and Modifying a Data Record

In this example, the data set to be modified is a volume table of contents.

```
//ZAPIT        JOB        MSGLEVEL=(1,1)
//STEP         EXEC       PGM=AMASPZAP
//SYSPRINT     DD         SYSOUT=A
//SYSLIB       DD         DSNAME=FORMAT4.DSCB,DISP=OLD,
//      UNIT=3330,VOLUME=SER=111111,DCB=(KEYLEN=44)
//SYSIN        DD         *
 CCHHR            0005000001
 VERIFY           2C   0504
 REP              2C   0A08
 REP              2E   0001,03000102
 ABSDUMPT         ALL
/*
```

SYSPRINT DD Statement: Defines the message data set.

SYSLIB DD Statement: Defines the data set to be accessed by SPZAP in performing the operations specified by the control statements. In this example, it defines the VTOC (a Format 4 DSCB) on a 3330 volume with a serial number of 111111. DCB=(KEYLEN=44) is specified so that the dump produced by the ABSDUMPT control statement will show the dsname which is a 44-byte key. Note that this is not necessary for the VERIFY and REP control statements.

CCHHR Control Statement: Indicates that SPZAP is to access the direct access record address "0005000001" in the data set defined by the SYSLIB DD statement while performing the operations specified by the following control statements.

VERIFY Control Statement: Requests that SPZAP check the data at hexadecimal displacement X'2C' from the start of the data record defined in the CCHHR statement to make sure it is the same as the hexadecimal data specified in this control statement. If the data is the same, SPZAP continues processing the following control statements sequentially. If the data is not identical, SPZAP does not perform the REP function but does perform the ABSDUMPT operation; it also dumps a formatted hexadecimal image of the data record defined by the CCHHR statement to the SYSPRINT data set.

REP Control Statements: Cause the eight bytes of data starting at displacement 2C from the beginning of the record to be replaced with the hexadecimal data in the REP control statements. The 2C displacement value allows for a 44-byte key at the beginning of the record.

ABSDUMPT Control Statement: Causes SPZAP to dump the entire data set to the SYSPRINT data set. Since DCB=(KEYLEN=44) is specified on the SYSLIB DD statement, the 44-byte dsname is also dumped.

Note: If the VTOC is to be modified, message AMA117D is to be issued to the operator, requesting permission for the modification.

Example: Using SPZAP to Modify a Record

This example shows how to inspect and modify a record within a PDSE data library.

```
//UPDDATA JOB   MSGLEVEL=(1,1)
//ZAPSTEP EXEC  PGM=AMASPZAP
//SYSPRINT  DD  SYSOUT=A
//SYSLIB    DD  DSN=IBMUSER.LMD.PDSE,DISP=OLD
//SYSIN     DD  *
  NAME    GETCSECT
  RECORD    0003
  VER       000010   47F0,F016
  REP       000014   10C7,C5E3,C4E2
  ABSDUMP   3        3
```

SYSLIB DD statement: Defines the data set that SPZAP is to access to perform the operations specified by the control statements. In this example, it defines a private PDSE data library. The NAME statement identifies the member as GETCSECT, which is shown in Figure 16-8 on page 16-34.

SYSIN DD statement: Defines the input stream containing the SPZAP control statements.

NAME control statement: Instructs SPZAP that the control statements that immediately follow this statement are to be performed on the member whose name is GETCSECT.

RECORD control statement: Indicates that SPZAP is to access relative record 3, the third record in the member GETCSECT. Record 3 is the object of the VERIFY and REP operations that follow.

VERIFY control statement: Requests that SPZAP check the data at hexadecimal displacement X'0010' to compare it to the string in the following REP statement. If there is a difference, this VERIFY is flagged with an error message, the contents of record 3 are displayed, and the following REP statement is flagged and ignored.

REP control statement: Causes SPZAP to replace the data at offset X'000014' in record 3 of member GETCSECT with the data X'10C7C5E3C4E2' if the preceding VERIFY statement completed successfully. If the preceding VERIFY statement was flagged in error, then this statement is also flagged in error, and no data is replaced.

ABSDUMP control statement: Causes SPZAP to display record 3 of member GETCSECT. Record 3 is displayed whether the VERIFY succeeded or failed.

Accessing a Data Record

The CCHHR control statement can only be used if SYSLIB is a non-PDSE data set. When you use the CCHHR control statement, SPZAP reads directly the physical record you want to inspect or modify. The offset parameters specified in subsequent VERIFY and REP statements are then used to locate the data that is to be verified or replaced within the record. These hexadecimal offsets must define the displacement of data relative to the beginning of the record and include the length of any key field.

To access data records in a PDSE, you must specify a NAME control statement. Any attempt to access records in a PDSE with a CCHHR control statement will cause an error message. Any VER|VERIFY, REP, IDRDATA, or SETSSI control statements immediately following a CCHHR statement will be flagged in error and ignored.

Data records in a PDSE can be accessed only with the following pair of control statements:

```
NAME membername
RECORD nn...n
```

You cannot use these control statements to access program objects.

Updating the System Status Index (SSI)

You can use the SETSSI control statement to overlay the existing data in the SSI with your own data. For a complete description of the SETSSI control statement, see “SPZAP Control Statements” on page 16-17.

The SSI is a 4-byte field created by the linkage editor in the directory entry of a load module. It is useful for keeping track of any modifications that are performed on a load module. SPZAP updates the system status index automatically whenever it replaces data in the associated module.

Not all load modules have system status information. In those that do, the SSI is located in the last four bytes of the user data field in the directory entry. Figure 16-2 shows the position of the SSI in load module directory entries.

```
┌────────────────┬────────────┬───────┬─────────────────────┬────────────┐
* Member Name    *    TTR     *    C  *  User Data Field    *  SSI       *
* 1            8 *  9     11  *   12  *  13 to 70 maximum   *  variable  *
└────────────────┴────────────┴───────┴─────────────────────┴────────────┘
```

Figure 16-2. SSI Bytes in a Load Module Directory Entry

Figure 16-3 shows the composition of the SSI field and the flag bits used to indicate the types of changes made to the corresponding load module program.

The first byte of SSI information contains the member’s change level. When a load module is initially released by IBM, its change level is set at one. Thereafter, the change level is increased by one for each release that includes a new version of that program. If you make a change to the SSI for any of the IBM-released programs, take care not to destroy this maintenance level indicator unless you purposely mean to do so. To keep the change level byte at its original value, find out what information is contained in the SSI before using the SETSSI function. The LISTLOAD control statement of the LIST service aid can give you the information you need.

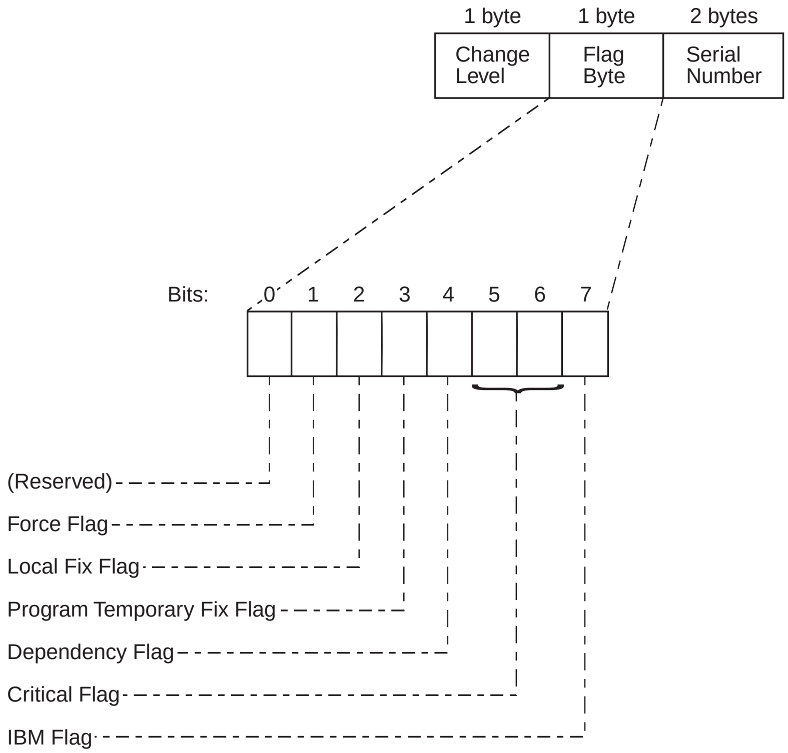

Figure 16-3. Flag Bytes in the System Status Index Field

The second byte of the SSI is called the *flag byte*. Bits within the flag byte contain information reflecting the member's maintenance status. You need only be concerned with two of the eight bits when you are using SPZAP:

- Bit 2, the local fix flag, indicates that the user has modified a particular member. (It is not used to reflect modifications made by IBM-supplied program temporary fix or a PTF.) SPZAP sets this local fix flag bit to one after successfully modifying a load module.
- Bit 3, the program temporary fix flag, is set to one when an IBM-authorized PTF is applied to a system library to correct an error in an IBM module.

All other bits in the flag byte should be retained in the SSI as they appeared before the SETSSI operation took place, so as not to interfere with the normal system maintenance procedures.

The third and fourth bytes of the system status index are used to store a serial number that identifies the first digit and the last three digits of a PTF number. SPZAP will not change these bytes unless you request a change by using the SETSSI control statement.

Running SPZAP

You can run SPZAP using control statements as input into the job stream or dynamically as part of selected macros:

- "Using JCL and Control Statements to Run SPZAP" on page 16-15
- "Invoking SPZAP Dynamically" on page 16-28

Operational Considerations

Consider the following points when you run SPZAP:

- SPZAP cannot inspect, modify, or dump data in partitioned data sets extended (PDSEs); PDSEs have a data structure that is different from that of partitioned data sets (PDSs).

  Reference

  See *Using Data Sets* for more information about PDSEs and their data structure.
- SPZAP uses the system OPEN macro. Therefore, SPZAP cannot modify or inspect RACF-protected data sets when SPZAP cannot successfully complete the access checks that occur during OPEN processing.
- A module can be a load module in a PDS or a program object in a PDSE. SPZAP replaces a program object in a PDSE rather than updating the program object in place. Users who have used the BLDL macro to establish a connection to a particular copy of a program object must invoke BLDL again to gain access to the new copy.

  If you are using LLA to manage a program object that has been changed through the use of SPZAP, then, to make the modified object available, the operator must refresh LLA for the directory entries for that program object. Otherwise, LLA continues to load the unmodified version of the program object.

  SPZAP itself cannot identify when a load module or a program object is in use by another user or is in the process of being loaded through LLA.
- Unexpired data sets such as system libraries cannot be modified unless the operator replies r xx,‘U’ to the expiration message that occurs during OPEN.
- If you use SPZAP to modify an operating system module that is made resident in virtual storage only at IPL time, you must IPL the system again to invoke the new version of the module you have modified. (Note that this requirement applies to all modules in SYS1.LPALIB.)

  SPZAP itself cannot determine when a module is loaded only at IPL time.
- The SYSLIB DD statement cannot define a concatenated or a multi-volume data set.
- SPZAP supports only direct access storage devices (DASD) for the SYSLIB device.
- When modifying a system data set, such as SYS1.LINKLIB, specify DISP=OLD on the SYSLIB DD statement.

Using JCL and Control Statements to Run SPZAP

One way to invoke SPZAP is through the job stream. The JCL statements you need to use when running SPZAP are:
- JOB statement
- EXEC statement
- SYSPRINT DD statement
- SYSLIB DD statement
- SYSABEND DD statement
- SYSIN DD statement

These JCL statements, when used with the control statements in “SPZAP Control Statements” on page 16-17, allow greater function for SPZAP.

Also, when running SPZAP, you must consider the region size available to your program. The minimum region size needed to run AMASPZAP is 17 kilobytes plus the larger of:
- 3 kilobytes
- The blocksize in bytes for the data set specified on the SYSLIB DD statement.

However, if you are running SPZAP at the DFSMS/MVS 1.1.0 level or later, SPZAP requires a minimum region size of 200K, specified either explicitly or implicitly. Normally, no REGION parameter is required on the EXEC statement but REGION=120K (or any other value less than 200K) will cause SPZAP to issue message AMA154I and stop processing with a return code of 16. In addition, SPZAP will issue message AMA154I if the program management binder has too small a region size. This problem might occur if the SYSLIB member is extremely large, when REGION=4M or REGION=6M might be needed.

JCL Statements

JOB Statement
Marks the beginning of the job.

EXEC Statement
Invokes SPZAP. You identify AMASPZAP as the program to be run by specifying either PGM=AMASPZAP or PGM=IMASPZAP, which is an alias name for AMASPZAP.

Note: You must be aware of the level of DFP or DFSMS that your system is running. If you are running DFSMS/MVS 1.1.0 or later, you must ensure that the region size is at least 200K for SPZAP to complete processing normally.

The only valid parameter that you may specify is PARM=IGNIDRFULL, which enables SPZAP to override the standard restrictions placed upon CSECT updates (via NAME and REP) when IDR space for the module is found to be full.

Notes:

1. Do not use PARM=IGNIDRFULL with IBM-maintained modules.
2. PARM=IGNIDRFULL has no meaning if SYSLIB is a program object library. There is no restriction on the number of IDRZ records associated with a program object library member.

SYSPRINT DD Statement
Defines a sequential output data set for messages that can be written on a system printer, a magnetic tape volume, or a direct access volume. This statement is required for each run of SPZAP.

SYSLIB DD Statement
Defines the direct access data set that will be accessed by SPZAP when performing the operations specified on the control statements. The DSNAME parameter and DISP=OLD or DISP=SHR are required. The VOLUME and UNIT parameters are necessary only if the data set is not cataloged. This statement cannot define a concatenated or multi-volume data set. It is required to run SPZAP.

Notes:

1. When this data set is the VTOC, you must specify DSNAME=FORMAT4.DSCB. When you access a record in the VTOC (that is, a DSCB) for modification, SPZAP issues message AMA117D to the console. No message is issued, however, when an ABSDUMPT operation is performed on the VTOC.
2. When you access a VSAM object (for example, rebuilding a catalog), you are required to reference the appropriate catalog on the STEPCAT or JOBCAT statement. If you fail to include a statement referring to the appropriate user catalog, your job might fail. If it does, your job is assigned

an abend 913-C, the data set is dumped, and the system displays message IEC150I. Note that using STEPCAT or JOBCAT statements for an SMS-managed data set, such as a PDSE, causes a JCL error.

Users of VSAM are required to use an ACB to open the VSAM data set. Because SPZAP only supports open with DCB, it does not obtain the correct information needed to operate upon a VSAM data set. Where there is a blocksize mismatch reported by SPZAP, explicit specification of the blocksize on the SYSLIB DD statement will override SPZAP's normal size processing.

SYSABEND DD Statement
Defines a sequential output data set to be used in case SPZAP ends abnormally. The data set can be written to a printer, a magnetic tape volume, or a direct access volume. This statement is optional.

SYSIN DD Statement
Defines the input stream data set that contains SPZAP control statements.

SPZAP Control Statements

SPZAP control statements (entered either through the user's input stream in the JCL or through the system console) define the processing functions to be performed during a particular run of SPZAP.

To enter other SPZAP control statements through the system console, you can use the CONSOLE control statement.

The control statements that define the running of SPZAP are:
- ABSDUMP or ABSDUMPT
- BASE
- CCHHR
- CHECKSUM
- Comment (*)
- CONSOLE
- DUMP or DUMPT
- IDRDATA
- NAME
- RECORD
- REP
- SETSSI
- VERIFY

Coding Rules for SPZAP Control Statements: Follow these rules when coding the control statements for SPZAP:
- The size of a SPZAP control card is 80 bytes; it can contain 71 bytes of control information.
- Statements can begin in any column up to column 71.
- The operation name of the statement must precede the parameters and must be complete on the first statement; you cannot continue the operation name.
- There must be at least one blank between the specified operation name and the first parameter.
- All parameters must also be separated by at least one blank space.
- Data field parameters may be formatted with commas for easier visual check, but blanks within data fields are not permitted.
- Data and offset values must be specified as a multiple of two hexadecimal digits.

- Following the last required parameter and its blank delimiter, the rest of the space on most control statements can be used for comments. Exceptions to this are the NAME and DUMP control statements: if you omit the CSECT parameter from either of these statements, do not use the space following the load module parameter for comments.
- A record beginning with an asterisk is considered to be a comment statement.
- A comment statement (one that begins with a single asterisk) cannot be continued.
- Member names and CSECT names for program objects can be as long as 1024 characters.
- When SYSLIB refers to a PDSE or HFS data set, you can continue any non-comment statement as follows:
  - Column 72 of the control card to be continued must contain a non-blank character.
  - The string of characters on the immediately following card (starting with the first non-blank character) is concatenated with column 71 of the preceding card. AMASPZAP ignores leading blanks in a continuation card, but it displays the cards on SYSPRINT unchanged.
  - You can continue statements as necessary. You cannot, however, continue a comment field that follows the last parameter.
  - Even though some parameters allow you to use a single asterisk (*) to indicate an omitted parameter, the first non-blank character on a continuation card cannot be an asterisk. Select the break point carefully to avoid starting a continuation statement with a single asterisk.

Following are detailed descriptions of the SPZAP control statements, in alphabetical order.

{ABSDUMP|ABSDUMPT}{startaddr stopaddr | startrec stoprec | membername | ALL}

This statement can be used to dump the following, as defined in the SYSLIB DD statement:
- A group of physical records
- A group of records belonging to a member of a data PDSE
- A load module member or all load module members of a PDS
- All members of a PDSE
- The directory of a PDSE that contains program objects

If the key associated with each record is to be formatted, DCB=(KEYLEN=nn), where “nn” is the length of the record key, must also be specified by the SYSLIB DD statement. Note that when dumping a VTOC, DCB=(KEYLEN=44) should be specified; when dumping a PDS directory, DCB=(KEYLEN=8) should be specified. ABSDUMP produces a hexadecimal printout only, while ABSDUMPT prints the hexadecimal data, the EBCDIC translation, and the mnemonic equivalent of the data (see “Reading SPZAP Output” on page 16-29). The variables are:

startaddr

The absolute direct access device address of the first record to be dumped. This address must be specified in hexadecimal in the form *cccchhhhrr* (cylinder, track and record numbers). This parameter must be exactly 10 digits long.

stopaddr
The absolute direct access device address of the last record to be dumped, and it must be in the same format as the start address.

Both addresses must be specified when this method of dumping records is used, and both addresses must be within the limits of the data set defined by the SYSLIB DD statement. The record number specified in the start address must be a valid record number. If a record number of 0 is specified, SPZAP will change it to 1 since the READ routine skips over such records. The record number specified as the stop address need not be a valid record number, but if it is not, the dump will continue until the last record on the track specified in the stop address has been dumped.

startrec
The value of the first relative record of a member of a PDSE data library to display. This parameter can be 1 to 8 digits long. The first record of a member has a *startrec* value of 1.

Note: *ABSDUMP|ABSDUMPT startrec stoprec* is valid only following a *NAME member* statement where SYSLIB is a PDSE data library and *member* is a valid member of that library.

stoprec
The value of the last relative record of a member of a PDSE data library to display. This parameter can be 1 to 8 digits long. If the value of *stoprec* specifies a relative record value greater than that of the last physical record, printing stops after the last record of the member is printed. If the value of *stoprec* is less than the value of *startrec*, no records are displayed. One can display all the records of a member of a PDSE data library by using the following two statements:

```
NAME  member
ABSDUMP|ABSDUMPT   1 99999999
```

membername
The name of a member of a PDS or a PDSE, as specified by the SYSLIB DD statement. The name can refer to a load module member of a PDS or a member of a PDSE data library. In each case, the entire member is dumped when this variable is specified. (Use DUMP/DUMPT for program object members of a PDSE.)

ALL
Specifies that the entire data set defined by the SYSLIB DD statement is to be dumped. How much of the space allocated to the data set is dumped depends on how the data set is organized:

- For a sequential data set, SPZAP dumps until it reaches end of file.
- For an indexed sequential and direct access data set, SPZAP dumps all extents.
- For a PDS, SPZAP dumps all extents, including all linkage editor control records, if any exist.
- For a PDSE data library, SPZAP displays a directory plus a listing of all members of the library. If the data set is a PDSE that contains program objects, SPZAP displays only the directory.

BASE xxxxxx
Used by SPZAP to adjust offset values that are to be specified in any subsequent VERIFY and REP statements. This statement should be used when

the offsets given in the VERIFY and REP statements for a CSECT are to be obtained from an assembly listing in which the starting address of the CSECT is not location zero.

For example, assume that CSECT ABC begins at assembly listing location X'000400', and that the data to be replaced in this CSECT is at location X'000408'. The actual displacement of the data in the CSECT is X'08'. However, an offset of X'0408' (obtained from the assembly listing location X'000408') can be specified in the REP statement if a BASE statement specifying X'000400' is included prior to the REP statement in the SPZAP input stream. When SPZAP processes the REP statement, the base value X'000400' will be subtracted from the offset X'0408' to determine the proper displacement of data within the CSECT. The variable is:

xxxxxx
A 6-character hexadecimal offset that is to be used as a base for subsequent VERIFY and REP operations. This value should reflect the starting assembly listing address of the CSECT being inspected or modified.

Note: The BASE statement should be included in the SPZAP input stream immediately following the NAME statement that identifies the control section that is to be involved in the SPZAP operations. The specified base value remains in effect until all VERIFY, REP, and SETSSI operations for the CSECT have been processed.

Example: Using the BASE Control Statement

This example shows how to use the BASE control statement to inspect and modify a CSECT whose starting address does not coincide with assembly listing location zero.

```
//MODIFY     JOB          MSGLEVEL=(1,1)
//STEP       EXEC         PGM=AMASPZAP
//SYSPRINT   DD           SYSOUT=A
//SYSLIB     DD           DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSIN      DD           *
 NAME             IEFMCVOL  IEFCVOL2
 BASE            0398
 IDRDATA         MOD04
 VERIFY          039A 5820C010
 REP             039A 47000000
 DUMP            IEFMCVOL   IEFCVOL2
```

SYSLIB DD Statement: Defines the system library, SYS1.LINKLIB, that contains the module IEFMCVOL in which the CSECT to be changed, IEFCVOL2, resides.

SYSIN DD Statement: Defines the input stream that contains the SPZAP control statements.

NAME Control Statement: Instructs SPZAP that the operations defined by the control statements that immediately follow it are to be performed on CSECT IEFCVOL2 in the load module IEFMCVOL.

BASE Control Statement: Provides SPZAP with a base value that is to be used to readjust the offsets on the VERIFY and REP statements that follow it.

IDRDATA Control Statement: Causes SPZAP to update the IDR in module IEFMCVOL for CSECT IEFCVOL2 with the data MOD04, if the REP operation is successful.

VERIFY Control Statement: Requests that SPZAP inspect the data at offset X'039A'. The base value X'0398' given in the previous BASE statement is subtracted from this offset to determine the proper displacement of the data within CSECT IEFCVOL2. Therefore, SPZAP checks the data at the location that is actually displaced X'0002' bytes from the beginning of CSECT IEFCVOL2 to ensure that it is the same as the hexadecimal data specified in this control statement. If the data is the same, SPZAP continues processing the following statements in the order in which they are encountered. If the data is not identical, SPZAP does not perform the REP, SETSSI, or IDRDATA functions, but it does perform the DUMPs operation; it also dumps a hexadecimal image of CSECT IEFCVOL2 to the SYSPRINT data set.

REP Control Statement: Causes SPZAP to replace the data at displacement X'0002' (offset 039A minus base value 0398) into CSECT IEFCVOL2 with the hexadecimal data specified in this control statement.

DUMP Control Statement: Requests that SPZAP dump a hexadecimal image of CSECT IEFCVOL2 to the SYSPRINT data set. Since the DUMP statement follows the REP statement, the image will reflect the changes made by SPZAP (assuming no verification has been rejected).

CCHHR record address

Identifies a physical record on a direct access device that is to be modified or verified. The record must be in the data set defined by the SYSLIB DD statement. Any immediately following REP or VERIFY statements will reference the data in the specified record. The variable is:

record address

The actual direct access address of the record containing data to be replaced or verified. It must be specified as a 10-digit hexadecimal number in the form *ccccHhhhrr*, where *cccc* is the cylinder, *hhhh* is the track, and *rr* is the record number. For example, 0001000A01 addresses record 1 of cylinder 1, track 10. A zero record number is incorrect and defaults to 1.

Note: You can define more than one CCHHR statement in your input to SPZAP. However, the VERIFY, REP and SETSSI statements associated with each CCHHR statement must immediately follow the specific CCHHR statement to which they apply.

CHECKSUM [hhhhhhhh]

Used to print or verify a fullword checksum (parity-check). All of the valid hexadecimal operands since the preceding CHECKSUM statement or SPZAP initialization are logically concatenated into a single string divided into fullwords, the sum of which is the checksum. For example, the string 12345678FACE produces the checksum 0D025678. Each CHECKSUM statement resets the accumulated checksum value to zeros.

The CHECKSUM statement is effective in detecting clerical errors that may occur when transcribing an SPZAP type of fix. CHECKSUM does not prevent errors; it only causes a message to be issued. By the time the CHECKSUM statement is processed, all prior replaces have been done.

hhhhhhhh

8 hexadecimal characters that are compared with the checksum. If the two values are equal, a message is written indicating that the checksum was correct and has been reset.

If the operand field is blank, a message is written giving the actual value of the checksum, and indicating that the checksum has been reset.

When the CHECKSUM control statement is provided with an incorrect operand, the REP and SETSSI statements processed already are not affected.

If the operand is not valid or is not equal to the checksum, a message is written indicating incorrect operand or checksum error. All subsequent REP and SETSSI statements are ignored until the next NAME or CCHHR statement is encountered. The results of previously processed statements are not affected.

*** (Comment)**

When the first non-blank character in a statement is an asterisk, SPZAP recognizes the statement as a comment, used to annotate the SPZAP input stream and output listing. You can specify the asterisk in any position, but at least one blank space must follow the asterisk.

You can include any number of comment statements in the input stream, but you cannot continue a comment statement. When SPZAP recognizes a comment, it writes the entire statement to the data set specified for SYSPRINT.

CONSOLE

Indicates that SPZAP control statements are to be entered through the system console.

When this statement is encountered in the input stream, the following message is written to the operator:

```
AMA116A  ENTER AMASPZAP CONTROL STATEMENT OR END
```

The operator may then enter in any valid SPZAP control statement conforming to the specifications described in the beginning of this control statement discussion. After each operator entry through the console is read, validated, and processed, the message is reissued, and additional input is accepted from the console until “END” is replied. SPZAP will then continue processing control statements from the input stream until an end-of-file condition is detected.

Notes:

1. You can enter control statements through the console in either uppercase or lowercase letters, but AMASPZAP does not fold lowercase input to uppercase.
2. You cannot continue a control statement entered through the console.

Example: Entering SPZAP Control Statements Through Console

This example shows how to enable SPZAP control statements to be entered through the console.

```
//CONSOLIN   JOB      MSGLEVEL=(1,1)
//STEP       EXEC     PGM=AMASPZAP
//SYSPRINT   DD       SYSOUT=A
//SYSLIB     DD       DSNAME=SYS1.LINKLIB,DISP=OLD
//SYSIN      DD       *
        CONSOLE
/*
```

SYSLIB DD Statement: Defines the data set that contains the module to be updated.

SYSIN DD Statement: Defines the input stream.

CONSOLE Control Statement: Indicates that SPZAP control statements are to be entered through the console.

{DUMP|DUMPT} member [csect | ALL | *] [class-name]

Dumps a specific control section or all control sections in a load module in a PDS or a program object in a PDSE. DUMP produces a hexadecimal printout only, while DUMPT prints the hexadecimal data, the EBCDIC translation, and the mnemonic equivalent of the data (see “Reading SPZAP Output” on page 16-29). The variables are:

member

The member name of the load module in a PDS or program object in a PDSE that contains the control section(s) to be dumped. (Note: This variable, ‘member’, must correspond to the name of a member of the PDS or PDSE that is defined by the SYSLIB DD statement.

csect | ALL | *

Defines the name of the particular control section that is to be dumped. To dump all the CSECTs of a load module in a PDS or a program object in a PDSE, specify “ALL” instead of the CSECT name. If you omit the variable entirely, or, for program objects only, code “*”, SPZAP assumes that you mean to dump only the first CSECT in the load module or program object.

If you specify a CSECT name that SPZAP does not find in the member, SPZAP dumps all of the CSECTs in the member.

Note: DUMP or DUMPT applied to a CSECT consisting only of space allocations (DS statements) will produce no output between the statement printback and the dump-completed message.

class-name
Indicates, for program objects only, the class of text that you want to dump. The default is B_text. Specifying B_*, C_*, or D_* causes SPZAP to dump all text classes beginning with the string that precedes the asterisk. If you want to omit the CSECT name and supply a class-name, code a single asterisk for the CSECT name followed by the class-name.

For information about the values you can specify for class name, see *z/OS DFSMS Program Management*.

Note: SPZAP does not fold lowercase input to uppercase; be sure to enter class-name in the correct case.

IDRDATA xxxxxxxx
Causes SPZAP to place up to eight bytes of user data into the SPZAP CSECT identification record of the load module; this is only done if a REP operation associated with a NAME statement is performed and the load module was processed by the linkage editor to include CSECT identification records. The variable is:

xxxxxxxx
Eight (or fewer) bytes of user data (with no embedded blanks) that are to be placed in the user data field of the SPZAP IDR of the named load module. If more than eight characters are in the variable field, only the first eight characters will be used.

Note: The IDRDATA statement is valid only when used in conjunction with the NAME statement. It must follow its associated NAME statement, or the BASE statement associated with a NAME statement, and precede any DUMP, DUMPT, ABSDUMP or ABSDUMPT statement. IDRDATA statements associated with CCHHR statements will be ignored.

NAME member [csect | *] [class-name]
Identifies a CSECT in a load module member of a PDS or a program object member of a PDSE or a program object member of a HFS data set that is to be the object of subsequent VERIFY, REP, SETSSI, or IDRDATA operations. The variables are:

member
The member name of the load module belonging to a PDS or the program object belonging to a PDSE or the program object belonging to a HFS data set that includes the CSECT that contains the data to be inspected or modified. The load module or the program object must be a member of the data set defined by the SYSLIB DD statement.

csect | *
The name of the particular control section that contains the data to be verified or replaced. If you omit this variable, or, for program objects only, code "*", SPZAP assumes that the first CSECT in the load module contained in a PDS or the program object contained in a PDSE or the program object contained in a HFS data set is the one to be used. If there is only one CSECT in the load module or program object, this variable is not necessary.

If you specify a CSECT name that SPZAP does not find in the member you name, then SPZAP does not perform any requested processing. Instead, it produces hexadecimal dumps of all CSECTs in the member. (The class of text dumped is specified on the class-name variable, and the default is B_text.)

class-name
Indicates, for program objects only, the class of text that you want to modify. The default is B_text. If you want to omit the CSECT name and supply a class-name, code a single asterisk for the CSECT name, followed by the class-name.

For information about the values you can specify for class name, see *z/OS DFSMS Program Management*.

Note: SPZAP does not fold lowercase input to uppercase; be sure to enter class-name in the correct case.

Note that you can define more than one NAME statement in your input to SPZAP. However, the VERIFY, REP and SETSSI statements associated with each NAME statement must immediately follow the NAME statement to which they apply. For a HFS program object, the member name may be the primary name or an alias with a maximum length of eight bytes.

NAME member
Identifies the member of a data library that is to be the object of subsequent VERIFY, REP, ABSDUMP, ABSDUMPT or RECORD operations. The variable is:

member
The member name of a data library whose contents are to be displayed, verified and/or replaced.

RECORD nnnnnnnn
This statement identifies a particular record in a member of a PDSE data library and must follow a *NAME member* statement where *member* specifies the name of the member. The combination of NAME and RECORD defines the record for which VER|VERIFY and possible REPs are to be performed.

nnnnnnnn consists of 1 to 8 decimal digits and specifies the relative record of interest. Leading zeroes are ignored. For example, the first record of a member may be specified as 1 or 01 or 00000001.

REP offset data
Modifies data at a specified location in a CSECT or physical record that was previously defined by the NAME, NAME/RECORD combination, or CCHHR statement. The data specified on the REP statement will replace the data at the record or CSECT location stipulated in the offset variable field.

SPZAP issues message AMA122I to record the contents of the specified location as they were before the change was made.

Note: IBM recommends that, before you replace any data, you always use VER/VERIFY to make sure that the contents you are going to change with the REP function are what you expect. The offset and length that you specify on the VER/VERIFY statement, however, do not need to match any following REP statement exactly; a single successful VERIFY can validate multiple following REP statements.

offset
Provides the hexadecimal displacement of data to be replaced in a CSECT or data record. This displacement need not address a fullword boundary, but it must be specified as a multiple of two hexadecimal digits (0D, 02C8, 001C52).

If the offset value is outside the limits of data record (physical block) or CSECT being modified, the replacement operation will not be performed.

When replacing data in a record with a key, the length of the key should be considered in the calculation of the displacement; that is, offset zero is the first byte of the key, not of the data.

data

Defines the bytes of data to be inserted at the location. As with the offset variable, the number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, the data within the variable may be separated by commas (never blanks); but again, the number of digits between commas must also be a multiple of two. For example, a REP data variable may look like this:

```
4160B820 (without commas)
    or like this:
4160,B820 (with commas).
```

If all the data to be modified does not fit into one REP statement (72 bytes), you can code another REP statement.

Notes:

1. Remember that SPZAP automatically updates the system status index (SSI) when it successfully modifies the CSECT named or implied on the previous NAME statement.
2. If you are performing multiple VERIFY and REP operations on a CSECT, make sure that all the VERIFY statements precede all the REP statements. This procedure ensures that all REP operations are ignored if one VERIFY reject occurs.
3. You are not required to supply a VERIFY statement before the first REP statement; however, when SPZAP encounters a VERIFY statement, it must be satisfied before SPZAP processes any following REP requests.
4. When you access a record in the VTOC (for example, the data set control block (DSCB)) for modification, SPZAP issues the message AMA117D to the console. No message is issued, however, when an ABSDUMPT operation is performed on the VTOC.

SETSSI xxyynnnn

Places user-supplied system status information in the directory entry for the load module member in a PDS or program object member in a PDSE. The SSI entry must have been created when the load module or program object member was link edited. The variable is:

xxyynnnn

Four bytes of system status information the user wishes to place in the SSI field for this member. Each byte is supplied as two hexadecimal digits indicating the following:

```
xx   - change level
yy   - flag byte
nnnn - modification serial number
```

If SPZAP detects an error in any previous VERIFY or REP operation, the SETSSI function is not performed.

Note: Because all bits in the SSI entry are set (reset) by the SETSSI statement, be very careful when using it to avoid altering the vital maintenance-status information. SPZAP issues message AMA122I to record the SSI as it was before the SETSSI operation was performed. (See “Updating the System Status Index (SSI)” on page 16-13.)

{VERIFY|VER} offset expected-content

Causes the data at a specified location within a CSECT or physical record to be compared with the data supplied in the statement.

offset

The hexadecimal displacement of data to be inspected in a CSECT or record.

This displacement does not have to be aligned on a fullword boundary, but it must be specified as a multiple of two hexadecimal digits, such as 0D, 021C, 014682. If this offset value is outside the limits of the CSECT or data record defined by the preceding NAME, NAME/RECORD, or CCHHR statement, the VERIFY statement is rejected. If this offset value plus the length of the expected-content string is outside the limits of the CSECT or record defined by the preceding NAME, NAME/RECORD combination, or CCHHR statement, the VERIFY statement is rejected and flagged in error. When inspecting a record with a key, the length of the key should also be considered in the calculation of the displacement; that is, offset zero is the first byte of the key.

expected-content

Defines the bytes of data that are expected at the specified location. As with the offset variable, the number of bytes of data defined must be specified as a multiple of two hexadecimal digits. If desired, the data within the parameter may be separated by commas (never blanks), but again, the number of digits between commas must also be a multiple of two. For example, expected content might look like this:

```
5840C032 (without commas),
or like this:
5840,C032 (with commas)
```

If all the data does not fit into one VERIFY statement (80-byte logical record), then another VERIFY statement must be defined.

Note: If the two fields being compared are not in agreement, that is, if the VERIFY operation is rejected, no succeeding REP or SETSSI operations are performed until the next NAME or CCHHR control statement is encountered. SPZAP provides a formatted dump of each CSECT or record for which a VERIFY operation failed.

Return Codes: When SPZAP ends, one of the following return codes is placed in general purpose register 15:

Code	Meaning
00	Successful completion.
04	Warning of a condition. This may result in future errors if an action is not taken to correct the warning now.
08	A SPZAP input statement contains an error or was overridden by operator intervention. Check the syntax of the statements to determine the cause of the error.
12	A requested JCL statement is absent or specifies a data set that was not successfully opened. SPZAP ends immediately.
16	A permanent I/O error has occurred, perhaps caused by a JCL error, such as incorrect blocksize. SPZAP ends immediately.

If you are running SPZAP at the DFSMS/MVS 1.1.0 level or later, the region size is too small. REGION=200K is the smallest permitted. However, the program management binder might require as much as 4M or 6M if the program object is very large.

20 Using DUMP, DUMPT, VER, or REP processing, SPZAP found a control record for a specific control section that was larger than the specified BLOCKSIZE. SPZAP ends immediately.

Invoking SPZAP Dynamically: You can run SPZAP from selected macros. SPZAP can be invoked by an application program at run time through the use of the CALL, LINK, XCTL, or ATTACH macro. The program must supply a list of alternate DDNAMEs of data sets to be used by SPZAP if the standard DDNAMEs are not used.

The general form of these macros when used to invoke SPZAP is shown below.

```
(anyname)   CALL SPZAP,(oplist,ddnamlst),VL
(anyname)   XCTL EP=SPZAP
(anyname)   LINK EP=SPZAP,PARAM=(oplist,ddnamlst),VL=1
(anyname)   ATTACH EP=SPZAP,PARAM=(oplist,ddnamlst),VL=1
```

anyname
Indicates an optional statement label on the macro statement.

SPZAP
is AMASPZAP or an alias.

EP
The entry point for the SPZAP program.

PARAM
Specifies, as a sublist, parameters to be passed from the program to SPZAP.

oplist
Specifies the name of either a halfword of zeros (indicating no options) or a non-zero halfword followed by a character string whose length is given in bytes. For the possible parameter value, see the information about the EXEC statement in “JCL Statements” on page 16-16.

ddnamlst
Specifies the name of a variable-length list containing alternate ddnames for data sets to be used during SPZAP processing. If all the standard ddnames (SYSPRINT, SYSLIB, and SYSIN) are used, then you can omit this parameter.

The DDNAME list must begin on a halfword boundary. The first two bytes contain a count of the number of bytes in the rest of the list. The format of the list is fixed, with each entry having eight bytes. Any name of less than eight bytes must be left justified and padded with blanks. If a name is left out in the list, the entry must contain binary zeros; the standard name is then assumed. Names can be omitted from the end of the ddname list by shortening the list.

The sequence of 8-byte entries in the list is as follows:

Entry	Standard name
0-7	not applicable
8-15	not applicable
16-23	not applicable
24-31	SYSLIB
32-39	SYSIN
40-47	SYSPRINT

VL | VL=1
Indicates that the high-order bit is to be set to 1 in the last word of the address parameter list.

Note: If you do not supply the name of a DDNAME list, you must ensure that the high-order bit of the oplist address is set on. Coding VL|VL=1 sets the bit correctly.

Figure 16-4 is an example of two functionally equivalent dynamic invocations of SPZAP.

```
         EXSPZAP  CSECT
         USING *,15           ASSUME REG15 IS BASE
         MODID                MODULE ID AND DATE IN PROLOG
         SAVE  (14,12)        SAVE REGISTERS
         BALR  12,0           ESTABLISH BASE REGISTER
         USING *,12
         ST    13,SAVEAREA+4  CHAIN NEW SAVEAREA TO PREVIOUS
         LR    2,13           TEMPORARILY SAVE ADDRESS OF OLD SAVEAREA
         LA    13,SAVEAREA    INIT REG13 WITH ADDRESS OF NEW SAVEAREA
         ST    13,8(0,2)      CHAIN PREVIOUS SAVEAREA TO NEW
*************************************************************************
*                                                                       *
*   THIS EXAMPLE SHOWS TWO FUNCTIONALLY EQUIVALENT DYNAMIC              *
*   INVOCATIONS OF SUPERZAP.                                            *
*           NO OPTIONS ARE PASSED.                                      *
*           THE DDNAME FOR THE SYSLIB FILE IS CHANGED TO TESTLIBR.      *
*           THE DDNAME FOR THE SYSIN FILE IS NOT CHANGED.               *
*           THE DDNAME FOR THE SYSPRINT FILE IS CHANGED TO PRINTOUT.    *
*                                                                       *
*************************************************************************
LINKZAP1 LINK  EP=AMASPZAP,PARAM=(OPTLIST,DDLIST),VL=1
LINKZAP2 LINK  EP=AMASPZAP,PARAM=(0,DDLIST),VL=1
         L     13,SAVEAREA+4  LOAD ADDRESS OF PREVIOUS SAVEAREA
         RETURN (14,12),T,RC=0 RETURN TO CALLER
OPTLIST  DC    H'0'           NO OPTIONS ARE PASSED TO AMASPZAP
DDLIST   DS    0H             ALIGN DDNAMES TO HALFWORD BOUNDARY
         DC    H'48'          LENGTH OF THE CHARACTER STRING
*                              CONTAINING DDNAME OVERRIDES
         DC    24XL1'00'      FIRST 24 CHARACTERS ARE IGNORED
         DC    CL8'TESTLIBR'  CHANGE SYSLIB FILE TO DDNAME OF TESTLIBR
         DC    8XL1'00'       USE SYSIN FILE FOR INPUT OF CONTROL
*                              STATEMENTS
         DC    CL8'PRINTOUT'  CHANGE SYSPRINT FILE TO DDNAME OF
*                              PRINTOUT
SAVEAREA DC    18F'0'         REGISTER SAVEAREA
         END
```

Figure 16-4. Sample Assembler Code for Dynamic Invocation of SPZAP

Reading SPZAP Output: SPZAP provides two different dump formats for the purpose of checking the data that has been verified or replaced. These dumps (written to the SYSPRINT data set specified by the user) may be of the formatted hexadecimal type or the translated form. Both formats are discussed below in detail with examples showing how each type will look.

Formatted Hexadecimal Dump: When DUMP or ABSDUMP is the control statement used, the resulting printout is a hexadecimal representation of the requested data. Figure 16-5 on page 16-31 gives a sample of the formatted hexadecimal dump. A heading line is printed at the beginning of each block. This heading consists of the hexadecimal direct access address of the block (ABSDUMP only), the length of the record, the class of text (program objects only), and the names of the member and the CSECT that contain the data being printed (if the dump is for specific CSECT or load module). Each printed line thereafter has a three-byte displacement address at the left, followed by eight groups of four data bytes each. The following message is printed under the last line of the dump printout:

```
AMA113I  COMPLETED DUMP REQUIREMENTS
```

Translated Dump: The control statements DUMPT and ABSDUMPT also provide an operation code translation and an EBCDIC representation of the data contained in the dump. Figure 16-6 on page 16-31 shows the format of the translated dump. The first byte of each halfword of data is translated into its mnemonic operation code equivalent, provided such a translation is possible. If there is not equivalent mnemonic representational value to be given, the space is left blank. This translated line of codes and blanks is printed directly under the corresponding hexadecimal line. An EBCDIC representation of each byte of data is printed on two lines to the right of the corresponding line of text, with periods substituted for those bytes that do not translate to valid printable characters.

```
DUMP    IEHMVESN   ALL

**CCHHR- 0022001108   RECORD  LENGTH-  0850              MEMBER NAME  IEHMVESN  CSECT NAME   IEHMVSSN
  000000     47F0F014     0EC5E2D5     60E6D9C1     D760E4D7        60606000      90ECD00C      189F5010      D0484110
  000020     D04850D0     10045010     D00818D1     5810D000        9200D00C      92FFD008      9140C20A      4780904A
  000040     9200C2F4     D20EC2F5     C2F49108     C20C4710        90E69500      C2FC4780      9064D203      C3009664
  000060     9200C2FC     D203C320     C31C95FF     C32A4770        908A4180      C00141F0      001450E0      964845E0
  000080     951858EO     96484520     95705820     C2640700        45109098      00000000      50210000      92801000
  0000A0     0A1495FF     C3274780     910A9108     C20C4710        91685820      C2749581      20114770      90D09102
  0000C0     C2084710     90F89110     C2084710     90F80700        451090D8      00000000      50210000      92801000
  0000E0     0A1447F0     910A9180     C1FC4780     9168947F        C1FC47F0      908A0700      45109100      00000000
  000100     50210000     92B01000     0A1495FF     C3344780        96DC41A0      C0089200      C2F49200      C2F89200
  000120     C2FC9200     C30094F7     A0429101     C2094780        91689102      C2094710      91685810      C27458F0
  000140     10149601     101748E0     F0044CE0     F0069101        10204710      915E4100      E00847F0      91624100
  000160     E0104110     F0000A0A     1B444340     C2245810        C2245830      C27C4833      000E95FF      30024780
  000180     918CD505     30041004     47F09192     D505301C        10044780      91E84111      000C4640      917A4140
  0001A0     000C1B14     41400001     D2031000     301095FF        30024780      91C0D205      10043004      47F091C6
  0001C0     D2051004     301C1B33     403096FC     D201100A        96FC4130      00019580      10024780      91E24030
  0001E0     96FCD201     100A96FC     5010C224     4240C224        9110C208      47109204      9102C208      47109204
  000200     47F09236     5810C224     95801002     47709236        D20196FC      100A4820      96FC4122      00014130
  000220     00011932     4770922C     41220001     402096FC        D201100A      96FC9140      C2094710      92B85820
  000240     00105822     00284832     00005930     92B44780        92B81233      47809268      91203012      47809268
  000260     91023003     47109270     41220002     47F09246        D203C228      C2005820      C200D203      200030
  000280     D2052004         C4122    000C5020     C2009640        C20947F0                    C2004143      00

  000600     41F0C014     D205F000     100441FF     0006D201        41FF0005      41FF0001      4111000C      46009604
  000620     F0004EE0     C080F337     F001C080     96F0F004        58FF0000      D219C014      F0019200      C33C07FE
  000640     58E09648     07FE1BDD     7FFF0000     58F09660        D503C31C      97004780      96D89500      C3284780
  000660     00000708     04000668     41800668     1BF8189F        4770969A      96FFC334      07FE58B0      C32058F0
  000680     96C25881     00001288     478096D8     95801008        1BBB43B0      C32806B0      42B0C328      41F00008
  0006A0     1000D24F     F000B000     41BB0050     50B0C320        0A0AD707      C31CC31C      1BFF07FE      9600C334
  0006C0     07FE58B0     C31C4100     0280181B     41110000        58E09648      45209570      47F09112      8CA00000
  0006E0     4180C001     41F00018     50E09648     45E09518
  000700     00000000     43A0400B

**CCHHR- 0022001108   RECORD  LENGTH-  0850              MEMBER NAME  IEHMVESN  CSECT NAME   IEHMVSSN
  000000     00000724     0000073F     00000750     00000761        00000775      00000793      000007E6      19E4D5C9
  000020     E340D9C5     C340D6D9     40E4D5D3     C1C2C5D3        C5C440E3      C1D7C50F      C9C5C8F3      F6F1C940
  000040     C4C1E3C1     40E2C5E3     0F404040     40404040        40C4C1E3      C140E2C5      E312C3D6      D7C9C5C4
  000060     40E3D640     E5D6D3E4     D4C54DE2     5D1CD5D6        E340D4D6      E5C5C460      C3D6D7C9      C5C440E3
  000080     D640E5D6     D3E4D4C5     4DE25D51     C9C5C8F3        F3F1C940      E4E2C5D9      40D3C1C2      C5D3E240
  0000A0     C1D9C540     D5D6E340     D4D6E5C5     C461C3D6        D7C9C5C4      4B40D5D6      40E4E2C5      D940D3C1
  0000C0     C2C5D340     E3D9C1C3     D240C1D3     D3D6C3C1        E3C5C440      C6D6D940      C9D5D7E4      E34B66C9
  0000E0     C5C8F3F3     F5C940D7     C5D9D4C1     D5C5D5E3        40C961D6      40C5D9D9      D6D940E6      C8C9D3C5
  000100     40E6D9C9     E3C9D5C7     40E4E2C5     D940D6E4        E3D7E4E3      40E3D9C1      C9D3C5D9      40D3C1C2
  000120     C5D3E24E     40D5D640     D4D6D9C5     40D3C1C2        C5D3E240      E6C9D3D3      40C2C540      D7D9D6C3
  000140     C5E2E2C5     C44B58B0
HMA1131  COMPLETED DUMPB REQUIREMENTS
```

Figure 16-5. Sample Formatted Hexadecimal Dump

```
DUMPT IEANUC05 SUTFPL59

                                                                                                                 00000100
**CCHHR- 01AB000416   RECORD LENGTH- 000068         MEMBER NAME  IEANUC05  CSECT NAME  SUTFPL59                  00000200
                                                                                                                 00000300
 000000   47F0 F01C  16E2 E4E3  C6D7 D3F5  F940 F9F8     F1F0 F540  C8C2  C2F6 F6F0 F600  90EC D00C  *.00..SUTFPL50 98*00000400
          BC   SRP   OR              MVZ   CP   CP       MVO                              STM        *105 HBB6606.....*00000500
          18BF 41C0  BEFF 41F0  0000 5800  B064 18A1     50D0 A004  50A0  D008 98F1 D010  18DA B365  *.......0........*00000600
 000020   LR   LA    ICM  LA         L          LR       ST         ST         LM         LR   LXR   *&...&....1......*00000700
 000040   0014 B362  0048 ED22  A00C 0006  ED32 B00C     C00E 58D0  D004  98EC D00C 07FE  0000 0000  *................*00000800
               LTXR       ED--             ED--               L           LM        BCR              *................*00000900
 000060   0000 0048  0000 0048                                                                       *........        *00001000

AMA113I COMPLETED DUMP REQUIREMENTS                                                                              00001100

AMA100I AMASPZAP PROCESSING COMPLETED                                                                            00001200
                                                                                                                 00001300
```

Figure 16-6. Sample Translated Dump

Figure 16-7 on page 16-33 shows CSECT output (obtained through DUMP/DUMPT) for a program object module.

Notes:

1. There are no **CCHHR** values. The program management binder manages its own DASD storage and returns no physical location.
2. **RECORD LENGTH: indicates length of the CSECT or module, not the length of the physical record containing the CSECT or module.
3. Program management binder returns no text for named or unnamed common areas. The length of the common section will be indicated. Message AMA152I indicates that no text has been returned.
4. SPZAP displays MEMBER NAME and CSECT NAME on as many lines as necessary. The names can be as long as 1024 characters.
5. SPZAP labels common storage in a program object with the tag COMMON NAME instead of CSECT NAME. Named common displays that name. Unnamed common is flagged as $BLANK COMMON. Private code is displayed with the subheading CSECT NAME: $PRIVATE CODE.

```
IGWSPZAP  INSPECTS, MODIFIES, AND DUMPS CSECTS OR SPECIFIC DATA RECORDS ON DIRECT ACCESS STORAGE.
   DUMP     MAINRTN   ALL                                                01770000

 **RECORD LENGTH: 000000C0 CLASS: B_TEXT   MEMBER NAME: MAINRTN
                                            CSECT NAME: $PRIVATE CODE
 00000000  90ECD00C   05C050D0   C02241E0   C01E50E0     D00818DE   58D0D004   58E0D00C   980CD014
 00000020  07FE0000   00000000   00000000   00000000     00000000   00000000   00000000   00000000
 00000040  00000000   00000000   00000000   00000000     00000000   00000000   00000000   00000000
 00000060  00000000   00000000   00000000   E3C8C9E2     40C9E240   C140D4C5   E2E2C1C7   C540C4C5
 00000080  C6C9D5C5   C440C9D5   40C140E4   D5D5C1D4     C5C440C3   E2C5C3E3   40404040   40404040
 000000A0  40404040   40404040   40404040   40404040     40404040   40404040   40404040

 **RECORD LENGTH: 00000058 CLASS: B_TEXT   MEMBER NAME: MAINRTN
                                           COMMON NAME: AAAAAAAA
 AMA152I NO TEXT DATA FOR THIS SECTION

 **RECORD LENGTH: 00000108 CLASS: B_TEXT   MEMBER NAME: MAINRTN
                                            CSECT NAME: SUBRTN
 00000000  90ECD00C   05C050D0   C06241F0   C05E50FD     000818DF   4510C034   001E8000   D5D6E640
 00000020  C9D540E3   C8C540C3   C1D3D3C5   C440D9D6     E4E3C9D5   C5400000   FF800A23   58B0C0A6
 00000040  D24FB004   C0AA9200   B002D201   B000C0FA     184B1814   0A231BFF   58D0C062   98ECD00C
 00000060  07FE0000   00000001   00000001   00000001     00000001   00000001   00000001   00000001
 00000080  00000001   00000001   00000001   00000001     00000001   00000001   00000001   00000001
 000000A0  00000001   00000001   00000001   00000448     C7D9C5C5   E3C9D5C7   E240C6D9   D6D440E3
 000000C0  C8C540E4   D5C3D6D4   D4D6D540   40404040     40404040   40404040   40404040   40404040
 000000E0  40404040   40404040   40404040   40404040     40404040   40404040   40404040   40404040
 00000100  00500000   00000000

 **RECORD LENGTH: 00000058 CLASS: B_TEXT   MEMBER NAME: MAINRTN
                                           COMMON NAME: $BLANK COMMON
 AMA152I NO TEXT DATA FOR THIS SECTION

 **RECORD LENGTH: 00000168 CLASS: B_TEXT   MEMBER NAME: MAINRTN
                                            CSECT NAME: MAINRTN
 00000000  90ECD00C   05C050D0   C06241F0   C05E50F0     D00818DF   4140C0B2   18140A23   5850C0AA
 00000020  D24FC0B6   50001814   0A235850   C0AED24F     C0B65000   18140A23   58B0C0A6   D24FB004
 00000040  C10A9200   B002D201   B000C15E   184B1814     0A2358F0   C15A05EF   58D0C062   98ECD00C
 00000060  1BFF07FE   00000000   00000000   00000000     00000000   00000000   00000000   00000000
 00000080  00000000   00000000   00000000   00000000     00000000   00000000   00000000   00000000
 000000A0  00000000   00000000   00000000   00000168     8000022C   800002EC   00500000   C7D9C5C5
 000000C0  E3C9D5C7   E240C6D9   D6D440E3   C8C540C3     C1D3D3C9   D5C740D9   D6E4E3C9   D5C5E240
 000000E0  D4C1C9D5   40E2C5C3   E3C9D6D5   40404040     40404040   40404040   40404040   40404040
 00000100  40404040   40404040   40404040   00000000     C8C940C6   D9D6D440   E3C8C540   C3C1D3D3
 00000120  C5D9E240   C3D6D4D4   D6D540E2   C5C3E3C9     D6D54040   40404040   40404040   40404040
 00000140  40404040   40404040   40404040   40404040     40404040   40404040   40404040   40404040
 00000160  00000340   00500000

 **RECORD LENGTH: 000000C0 CLASS: B_TEXT   MEMBER NAME: MAINRTN
                                            CSECT NAME: $PRIVATE CODE
 00000000  90ECD00C   05C050D0   C02241E0   C01E50E0     D00818DE   58D0D004   58E0D00C   980CD014
 00000020  07FE0000   00000000   00000000   00000000     00000000   00000000   00000000   00000000
 00000040  00000000   00000000   00000000   00000000     00000000   00000000   00000000   00000000
 00000060  00000000   00000000   00000000   E3C8C9E2     40C9E240   C4C1E3C1   40C4C5C6   C9D5C5C4
 00000080  40C9D540   C1D5D6E3   C8C5D940   C3E2C5C3     E340E6C9   E3C840D5   D640D5C1   D4C54040
 000000A0  40404040   40404040   40404040   40404040     40404040   40404040   40404040

 AMA113I COMPLETED DUMP REQUIREMENTS
 AMA100I IGWSPZAP   PROCESSING COMPLETED
```

Figure 16-7. Sample Formatted Hexadecimal Dump for PDSE Program Object Module

Figure 16-8 shows output for a member of a PDSE data library. *ABSDUMPT 0001 0500* would have been preceded by a *NAME membername* statement (not shown).

Note: There are no **CCHHR** values. RECORD NUMBER: shows the 8 digit value of the relative record number of the member being printed. RECORD LENGTH: shows the length of the record, while MEMBER NAME: shows the member name as it appears on the *NAME membername* statement.

```
IGWSPZAP  INSPECTS, MODIFIES, AND DUMPS CSECTS OR SPECIFIC DATA RECORDS ON DIRECT ACCESS STORAGE.
  NAME     GETCSECT                                                     03520001
  ABSDUMPT 0001  0050                                                   03530001

**RECORD NUMBER: 0000000001, RECORD LENGTH: 000050        MEMBER NAME: GETCSECT
 000000   02C5 E2C4  4040 4040  4040 0030  4040 0001     C7C5 E3C3  E2C5 C3E3  0000 0000  0200 0530   *.ESD............*
                     STH  STH   STH        STH                                                 BALR   *GETCSECT........*
 000020   E7E3 D5C4  E2E3 D240  0200 0000  4040 4040     E2E3 C1D9  E3C4 4040  0200 0000  4040 4040   *XTNDSTK.........*
               CLC        MVC              STH  STH                      STH              STH  STH    *STARTD..........*
 000040   4040 4040  4040 4040  F0F2 F4F8  F0F0 F0F1                                                  *........02480001*
          STH  STH   STH  STH   SRP        SRP  SRP                                                   *................*

**RECORD NUMBER: 0000000002, RECORD LENGTH: 000050        MEMBER NAME: GETCSECT
 000000   02C5 E2C4  4040 4040  4040 0020  4040 0004     D7D9 D5E3  D3C9 D5C5  0200 0000  4040 4040   *.ESD............*
                     STH  STH   STH        STH           XC   CLC   MVZ  CLC              STH  STH    *PRNTLINE........*
 000020   D7C1 D4C1  F1F4 F0C9  0200 0000  4040 4040     4040 4040  4040 4040  4040 4040  4040 4040   *PAMA140I........*
          XC   NC    MVO  SRP              STH  STH      STH  STH   STH  STH   STH  STH   STH  STH    *................*
 000040   4040 4040  4040 4040  F0F2 F4F9  F0F0 F0F1                                                  *........02490001*
          STH  STH   STH  STH   SRP        SRP  SRP                                                   *................*

**RECORD NUMBER: 0000000003, RECORD LENGTH: 000050        MEMBER NAME: GETCSECT
 000000   02E3 E7E3  4000 0000  4040 0038  4040 0001     47F0 F016  10C7 C5E3  C3E2 C5C3  E340 40F9   *.TXT............*
                     STH        STH        STH           BC   SRP   LPR                        STH    *.00..GETCSECT..9*
 000020   F24B F2F4  F400 90EC  D00C 18CF  1FFF 43F0     C52C 1F00  BF07 C52D  581D 0008  1A01 58FD   *2.244..........0*
          PACK PACK       STM         LR   SLR  IC            SLR   ICM        L          AR   L      *E.....E.........*
 000040   0000 190F  47D0 C050  F0F2 F5F0  F0F0 F0F1                                                  *.......&02500001*;
               CR    BC         SRP        SRP  SRP                                                   *................*

         ....

**RECORD NUMBER: 0000000028, RECORD LENGTH: 000050        MEMBER NAME: GETCSECT
 000000   02D9 D3C4  4040 4040  4040 0020  4040 4040     0002 0001  1C00 003C  0003 0001  0C00 0468   *.RLD............*
               MVZ   STH  STH   STH        STH  STH                 MR                   BASSM SPM    *................*
 000020   0004 0001  0C00 046C  0005 0001  0C00 0470     4040 4040  4040 4040  4040 4040  4040 4040   *.......%........*
                    BASSM SPM             BASSM SPM      STH  STH   STH  STH   STH  STH   STH  STH    *................*
 000040   4040 4040  4040 4040  F0F2 F7F5  F0F0 F0F1                                                  *........02750001*
          STH  STH   STH  STH   SRP        SRP  SRP                                                   *................*

**RECORD NUMBER: 0000000029, RECORD LENGTH: 000050        MEMBER NAME: GETCSECT
 000000   02C5 D5C4  4040 4040  4040 4040  4040 4040     4040 4040  4040 4040  4040 4040  4040 4040   *.END............*
               CLC   STH  STH   STH  STH   STH  STH      STH  STH   STH  STH   STH  STH   STH  STH    *................*
 000020   F2F5 F6F6  F8F9 F6F2  F0F1 40F0  F2F0 F1F9     F2F2 F4F4  C37D D7D3  61C1 E27D  4040 F0F1   *2566896201.02019*
          PACK       ZAP        SRP  STH   PACK MVO      PACK            XC               STH  SRP    *2244C'PL/AS'..01*
 000040   F0F4 F9F2  F2F4 F440  F0F2 F7F6  F0F0 F0F1                                                  *0492244.02760001*
          SRP  CP    PACK       SRP        SRP  SRP                                                   *................*
AMA113I COMPLETED DUMP REQUIREMENTS
AMA100I IGWSPZAP   PROCESSING COMPLETED
```

Figure 16-8. Sample Translated Dump for PDSE Data Library

Chapter 17. Dump Suppression

Save time! Save money! Suppress those dumps!

The system requests dumps you might not need. To keep from using your system's resources on unneeded dumps, you should suppress them. The reasons for the unneeded dumps and ways to suppress them are:

- **Duplicate dumps:** The system can request a dump for a problem that recurs. The dump written when the problem first occurs can be used for diagnosis; additional dumps are unneeded. Also, sometimes a system can request several dumps for one instance of a problem. A recurring problem may have been diagnosed, but the fix not yet incorporated into the system.

  To eliminate the duplicate dumps, use dump analysis and elimination (DAE). See "Using DAE to Suppress Dumps".
- **Dumps for certain abend codes:** For some abend codes, the accompanying messages provide the needed problem data. To eliminate the dumps for these abend codes, use a SLIP command. See "Using a SLIP Command to Suppress Dumps" on page 17-11.
- **A dump for an abend in an application program:** if the dump is not needed. See "Using an ABEND Macro to Suppress Dumps" on page 17-12.
- **Dumps the installation decides are not needed:** If you decide that certain dumps are not needed, you can code a routine for an installation exit to suppress these dumps. See "Using Installation Exit Routines to Suppress Dumps" on page 17-12.

This chapter lists the ways that an expected dump can be suppressed, so that you can determine why you did not receive an intended dump. See "Determining Why a Dump Was Suppressed" on page 17-12.

Using DAE to Suppress Dumps

Dump analysis and elimination (DAE) suppresses dumps that match a dump you already have. Each time DAE suppresses a duplicate dump, the system does not collect data for the duplicate or write the duplicate to a data set. In this way, DAE can improve dump management by only dumping unique situations and by minimizing the number of dumps.

This topic describes suppressing dumps using DAE as follows:

- "Performing Dump Suppression" on page 17-2
  - "Managing Rapidly Recurring Dumps" on page 17-4
- "Planning for DAE Dump Suppression" on page 17-5
  - "Selecting or Creating an ADYSETxx Parmlib Member" on page 17-5
  - "Defining a DAE Data Set" on page 17-7
- "Accessing the DAE Data Set" on page 17-8
  - "Generating a Suppressed Dump" on page 17-9
- "Stopping, Starting, and Changing DAE" on page 17-10

References

- See *z/OS MVS Diagnosis: Reference* for symptoms and symptom strings.
- See *z/OS MVS Initialization and Tuning Reference* for the ADYSETxx and IEACMD00 parmlib members.

- See *z/OS MVS IPCS Commands* for the VERBEXIT DAEDATA subcommand.
- See *z/OS MVS System Data Set Definition* for the DAE data set.
- See *z/OS MVS Planning: Global Resource Serialization* for data set serialization.
- See *z/OS Security Server RACF Command Language Reference* to control access to data sets.

Performing Dump Suppression

To perform dump suppression, DAE builds a symptom string, if the data for it is available. If the symptom string contains the minimum problem data, DAE uses the symptom string to recognize a duplicate SVC dump or SYSMDUMP dump requested for a software error. When installation parameters request suppression, DAE suppresses the duplicate dump. The following describes DAE processing.

1. **DAE obtains problem data**. DAE receives the data in the system diagnostic work area (SDWA) or from values in a SYMREC parameter on the SDUMP or SDUMPX macro that requested the dump.
   - The ESTAE routine or the functional recovery routine (FRR) of the failing program supplies module-level information, such as the failing load module name and the failing CSECT name.
   - The system supplies system-level data, such as the abend and reason codes, the failing instruction, and the register/PSW difference.

     If the failing component does not supply the failing load module name or CSECT name, the system determines the name, if possible. In this case, the name may be IEANUC0x.
2. **DAE forms a symptom string.** DAE adds a descriptive keyword to each field of problem data to form a symptom. DAE forms MVS symptoms, rather than RETAIN® symptoms. DAE combines the symptoms for a requested dump into a symptom string.

   The following tables show the required and optional symptoms. SDWA field names are given for the symptoms the failing program must provide to enable dump suppression. The tables have both MVS and RETAIN symptoms so that you can relate the MVS symptoms DAE uses to the RETAIN symptoms you might use to search the RETAIN data base. An MVS symptom string must contain at least five symptoms that are not null. DAE places symptoms into strings in the order shown in the tables.

   Required symptoms are first and must be present.

Symptom	SDWA Field	MVS Keyword	RETAIN Keyword
Name of the failing load module	SDWAMODN	MOD/name	RIDS/name#L
Name of the failing CSECT	SDWACSCT	CSECT/name	RIDS/name

Optional symptoms must follow the required symptoms. DAE needs at least three of these optional symptoms to make a useful symptom string.

Symptom	SDWA Field	MVS Keyword	RETAIN Keyword
Product/component identifier with the component identifier base	SDWACID, SDWACIDB	PIDS/name	PIDS/name
System completion (abend) code		AB/S0hhh	AB/S0hhh

Symptom	SDWA Field	MVS Keyword	RETAIN Keyword
User completion (abend) code		AB/Udddd	AB/Udddd
Recovery routine name	SDWAREXN	REXN/name	RIDS/name#R
Failing instruction area		FI/area	VALU/Harea
PSW/register difference		REGS/hhhhh	REGS/hhhhh
Reason code, accompanying the abend code or from the REASON parameter of the macro that requests the dump		HRC1/nnnn	PRCS/nnnn
Subcomponent or module subfunction	SDWASC	SUB1/name	VALU/Cname

3. **DAE tries to match the symptom string from the dump to a symptom string for a previous dump** of the same type, that is, SVC dump or SYSMDUMP. When DAE finds a match, DAE considers the dump to be a duplicate.

   Previous symptom strings are kept by DAE in virtual storage. When DAE is started, usually during IPL, DAE selects from the DAE data set symptom strings that were active in the last 180 days; either the string was created for a unique dump within the last 180 days or its dump count was updated within the last 180 days. The selected symptom strings are placed in virtual storage.

   The systems in a sysplex can share the DAE data set to suppress duplicate dumps across the sysplex. While each system in a sysplex can use its own DAE data set, IBM recommends that systems in a sysplex share a DAE data set so that:

   - DAE can write a dump on one system and suppress duplicates on other systems in the sysplex.
   - Only one DAE data set is required, rather than a data set for each system.

   See “Defining a DAE Data Set” on page 17-7 for more information, including recommended names for the data set.

4. **DAE updates the symptom strings in storage and, when the dump is written to a dump data set, in the DAE data set**, if updating is requested.

   - For a unique symptom string, DAE adds a new record. The record contains the symptom string, the dates of the first and last occurrences, the incidence count for the number of occurrences, and the name of the system that provided the string.
   - For a duplicate symptom string, DAE updates the incidence count for the string, the last-occurrence date, and the name of the last system that found the string.

   If updating is requested, DAE examines the incoming dump requests against captured dumps. If the incoming dump’s symptom string matches any dump on the captured dump queue, it is suppressed. Updates are done when the DAE data set is updated.

   In a sysplex, changes to the in-storage strings of other systems are made after the shared DAE data set is updated. If an incident is occurring at about the same time on multiple systems, multiple dumps will be generated — but only one per system. Dumps on other systems are suppressed after one of the dumps is written, the DAE data set updated, and the updates propagated to the other systems.

If the system with the original dump fails before it writes the captured dump, the dump will not be suppressed the next time it is requested.

5. **DAE suppresses a duplicate dump**, if DAE is enabled for dump suppression.

   Note that, if you specify an ACTION of SVCD, TRDUMP, or NOSUP on a SLIP command, the command overrides DAE suppression and the system writes the dump. Also, dumps requested by the DUMP operator command are not eligible for suppression.

   When DAE does not suppress a dump, the symptom string is in the dump header; you can view it with the IPCS VERBEXIT DAEDATA subcommand. DAE also issues informational messages to indicate why the dump was not suppressed.

   DAE suppresses a dump when all of the following are true:

   - DAE located in the dump the minimum set of symptoms.
   - The symptom string for the dump matches a symptom string for a previous dump of the same type.
   - Either of the following is true:
     – The current ADYSETxx parmlib member specifies SUPPRESS for the type of dump being requested and the VRADAE key is present in the SDWA.
     – The current ADYSETxx parmlib member specifies SUPPRESSALL for the type of dump being requested and the VRANODAE key is absent from the SDWA.

   The following table shows the effect of the VRADAE and VRANODAE keys on dump suppression when SUPPRESS and SUPPRESSALL keywords are specified in the ADYSETxx parmlib member. For SUPPRESS, the VRANODAE key can be present or absent; the system does not check it. The table assumes that the symptom string from the dump has matched a previous symptom string.

ADYSETxx Option	VRADAE Key in SDWA	VRANODAE Key in SDWA	Dump Suppressed?
SUPPRESS	Yes	N/A	Yes
SUPPRESS	No	N/A	No
SUPPRESSALL	Yes	No	Yes
SUPPRESSALL	No	Yes	No
SUPPRESSALL	No	No	Yes
SUPPRESSALL	Yes	Yes	No

   The only way to ensure that a dump is **not** suppressed, regardless of the contents of the ADYSETxx parmlib member, is to specify the VRANODAE key in the SDWA.

Managing Rapidly Recurring Dumps

DAE can suppress rapidly recurring dumps automatically and the support staff does not need to be aware when a dump request recurs. However, a surge of dump requests could affect system performance, even though the dumps are suppressed. The surge could go unnoticed for hours. To help the support staff take actions to avoid impact to users, the system can notify you of high-frequency dump requests.

To obtain notification, add a NOTIFY parameter to the SVCDUMP statement on the ADYSETxx parmlib member to establish a threshold for notification. The SVCDUMP statement must also specify UPDATE. The default threshold is 3 dumps requested in 30 minutes for the same symptom string. The notification time is measured from completion or suppression of dumps, rather than from initiation of dumps.

The notification is made by the event notification facility (ENF). You can use an ENF exit to:
- Notify the support staff by a message or a signal to a beeper
- Use automation

Any program can receive the ENF signal. If active, the First Failure Support Technology™ (FFST™) issues a generic alert in response to this ENF signal.

If DAE is stopped and restarted, DAE begins counting dumps again to reach the threshold.

If each system has its own DAE data set, notification is for a system. If the systems of a sysplex share the DAE data set, notification is for the sysplex. For example, with a shared DAE data set, four dumps for the same symptom string on the same or different systems in 25 minutes would cause notification if the ADYSETxx parmlib member contains NOTIFY(4,25).

Note: The system in the sysplex that crosses the notification threshold is the system that does the notify.

Planning for DAE Dump Suppression

Planning for DAE dump suppression consists of tasks to be done before an initial program load (IPL). The system programmer performs the following tasks:
- Selecting or creating an ADYSETxx parmlib member
- Defining a DAE data set

Selecting or Creating an ADYSETxx Parmlib Member

Select or create an ADYSETxx parmlib member to be used at IPL. IBM supplies three ADYSETxx members:

- **ADYSET00**, which starts DAE and keeps 400® symptom strings in virtual storage. The IBM-supplied ADYSET00 member contains:

  ```
  DAE=START,RECORDS(400),
      SVCDUMP(MATCH,SUPPRESSALL,UPDATE,NOTIFY(3,30)),
      SYSMDUMP(MATCH,UPDATE)
  ```

  ADYSET00 does not suppress SYSMDUMP dumps because installation-provided programs deliberately request them. If desired, change the ADYSETxx member being used to suppress SYSMDUMP dumps.
- **ADYSET01**, which stops DAE processing. The IBM-supplied ADYSET01 member contains:

  ```
  DAE=STOP
  ```

  When using the DAE Display facility's TAKEDUMP (T) action in a sysplex where DAE is active, you must change the contents of ADYSET01 to:

  ```
  DAE=STOP,GLOBALSTOP
  ```
- **ADYSET02**, which contains the same parameters as ADYSET00.

Dump Suppression

The IBM-supplied IEACMD00 parmlib member issues a SET DAE=00 command, which activates ADYSET00 during IPL. If you do not want DAE to start during IPL, change IEACMD00 to specify SET DAE=01.

For a sysplex, IBM recommends that you use the same ADYSETxx parameter values in each system. To use the same values, use a shared SYS1.PARMLIB. If your installation does not share a SYS1.PARMLIB, make the ADYSETxx and IEACMDxx members in the SYS1.PARMLIB for each system identical. A shared ADYSETxx or identical ADYSETxx members should specify SHARE(DSN) to share the DAE data set.

IBM recommends that the ADYSETxx member specify SUPPRESSALL, which requests that dumps be suppressed even though the component or program did not request dump suppression with a VRADAE key in the system diagnostic work area (SDWA). SUPPRESSALL is useful because it allows more dumps to be eligible for suppression.

Example: An ADYSETxx Member for a System in a Sysplex

The systems in the sysplex share a DAE data set, SYS1.DAESHARE, so DAE can suppress a duplicate of a previous dump from any system. This member also specifies SUPPRESSALL.

```
DAE=START,RECORDS(400),
    SVCDUMP(MATCH,SUPPRESSALL,UPDATE,NOTIFY(3,30)),
    SYSMDUMP(MATCH,UPDATE)
    SHARE(DSN,OPTIONS),
    DSN(SYS1.DAESHARE)
```

The ADYSET00 member specifies RECORDS(400). If your system does not suppress a dump when the matching symptom string is in the DAE data set, you might need more than 400 records in storage; the IBM Support Center can advise you.

Changing Parmlib Members to Change DAE Processing: While the system is running, change the DAE data set or parameters for the dumps by creating a new ADYSETxx parmlib member. See "Changing DAE Processing in a Sysplex" on page 17-11 for the operator actions needed to change the parmlib member.

There is another benefit when all the systems in a sysplex are sharing the DAE data set. That is, after DAE is started on each system using an ADYSETxx member which at least contains SHARE(DSN). One operator command can set the DAE values to be the same on all systems. This is accomplished by issuing the SET DAE= command, for an ADYSEYxx member which includes the GLOBAL parameter. ALL systems sharing the DAE data set will be effected.

Example: An ADYSETxx Member with GLOBAL

The following ADYSET04 member changes the DAE data set being used on all systems to SYS1.DAESH2 and changes the dump options on all systems.

```
DAE=START,RECORDS(400),
    SVCDUMP(MATCH,SUPPRESSALL,UPDATE,NOTIFY(3,30)),
    SYSMDUMP(MATCH,UPDATE)
    SHARE(DSN,OPTIONS),
    DSN(SYS1.DAESH2)
    GLOBAL(DSN,OPTIONS)
```

None of the changes made using operator commands are kept across an IPL of a system. At IPL, each system will again use the member specified in IEACMD00 or the COMMNDxx member being used. To make the changes permanently effective, do one of the following:

- Make the changes in ADYSET00 and use IEACMD00.
- Make the changes in an ADYSETxx member and use an IEACMDxx member that specifies that ADYSETxx in the SET DAE=xx statement.

Defining a DAE Data Set

Define a DAE data set when defining system data sets. When the system is IPLed or if DAE is stopped and restarted, DAE should continue using the DAE data set previously used.

1. **Define the DAE data set in a DD statement.** Use the default name of SYS1.DAE for a single system; use a different name for a DAE data set shared by systems in a sysplex.

   Example: DAE Data Set for Single System

   The sample DD statement is for a DAE data set used by a single system.

   ```
   //DAE    DD  DSN=SYS1.DAE,DISP=(,CATLG),VOL=(,RETAIN,SER=SG2001),
   //            DCB=(RECFM=FB,LRECL=255,DSORG=PS,BLKSIZE=0),
   //            UNIT=3390,SPACE=(TRK,(6,2))
   ```

   In a sysplex, each system can have its own DAE data set, but IBM recommends that all systems in a sysplex share a DAE data set.

   Example: DAE Data Set Shared by Sysplex Systems

   The sample DD statement is for a DAE data set shared by the systems in a sysplex. The statement will catalog the DAE data set in the shared master catalog or in the master catalog on each system that uses it.

   ```
   //DAE    DD  DSN=SYS1.DAESHARE,DISP=(,CATLG),VOL=(,RETAIN,SER=SG1055),
   //            DCB=(RECFM=FB,LRECL=255,DSORG=PS,BLKSIZE=0),
   //            UNIT=3390,SPACE=(TRK,(12,2))
   ```

   If you manage your dumps with the hierarchical storage manager (HSM), consider using an HSM purge time of 180 days to correspond to the DAE record aging of 180 days.

2. **If the DAE data set is shared by systems in a sysplex, establish a sysplex group for the sharing systems.** The sysplex group allows the ADYSETxx parmlib settings and the symptom record updates to be shared across the sysplex.

   In the COUPLExx parmlib member for each sharing system, specify a CLASSDEF statement that assigns the system to a group for DAE. The name of a group member is its system name, which is specified in the SYSTEM parameter in the IEASYSxx parmlib member. The class name on the CLASSDEF statement must be DAE and the group name must be SYSDAE.

   COUPLE00 Member for a Sysplex System

   The CLASSDEF statement specifies that the system is a member of the group SYSDAE in class DAE.

   ```
   COUPLE SYSPLEX(PLEX1)
     .
     .
     .
   CLASSDEF CLASS(DAE) GROUP(SYSDAE)
   ```

3. **Provide DAE data set integrity through a serialization component**, such as global resource serialization.

   For a single system, the DAE data set is a local resource. The default DAE data set, SYS1.DAE, is defined as a local resource in the default global resource serialization resource name list (RNL). If you give the DAE data set another name, add the name to the SYSTEMS exclusion RNL to avoid contention when more than one system uses the same DAE data set name for physically different data sets.

   For systems in a sysplex, the shared DAE data set is a global resource. To make global resource serialization treat it as a global resource, do one of the following:

   - Give the DAE data set a name other than SYS1.DAE. For example, SYS1.DAESHARE.
   - If you use the name SYS1.DAE, delete the DAE data set entry from the default SYSTEMS exclusion RNL. The DAE data set entry is `SYSDSN SYS1.DAE`.

   For information, see *z/OS MVS Planning: Global Resource Serialization*.

4. **Control access to the DAE data set.** On a single system or on all systems sharing the DAE data set in a sysplex, use Resource Access Control Facility (RACF) to control access. Enter a RACF ADDSD command to define a data set profile for the DAE data set.

Accessing the DAE Data Set

A DAE data set that is used by one system or is shared by systems in a sysplex is accessed by:

- Invoking the IPCS DAE Display panel
- Generating a suppressed dump
- Editing the DAE data set

Invoking the IPCS DAE Display Panel

For the ways to invoke the panel, see IPCS option 3.5 in the *z/OS MVS IPCS User's Guide* . On the panel, you can:

- View the symptom strings the data set contains by entering:

The date of the dump,
The last date for the string,
The number of times the dump has been requested,
And the last system that requested the dump.

- Search the Entry list for symptoms, system names, dates, etc.
- Navigate through the sysplex dump directory (or whatever dump directory is active) for the symptom string.
- View the dump title for a symptom string.

Generating a Suppressed Dump

You may want to obtain a dump that is being suppressed. Perhaps the first dump was ignored and thrown out, but since then the dump has been requested often enough so that you would like to analyze the dump. Do the following to obtain the suppressed dump through the IPCS TAKEDUMP option:

1. Customize the TSO userid that will invoke the TAKEDUMP action. Make sure it:
   - has authority to issue an MVS operator SET command and, if DAE is active in a sysplex, the ROUTE command
   - has RACF UPDATE access to the DAE data set.
2. Ensure that the ADYSET01 member(s) contains DAE=STOP (or DAE=STOP,GLOBALSTOP in a sysplex).
3. Check that the active IKJTSOxx member includes the program name ADYOPCMD in the AUTHCMD NAMES section.
4. In a sysplex, the maximum benefit is realized when DAE is started using ADYSETxx members which contain at least SHARE(DSN) — enabling shared data set activities.
5. Use the IPCS DAE dialog Panel to issue action code T (the TAKEDUMP option) on the line showing the symptom string of interest.

To process the TAKEDUMP option of the IPCS DAE dialog, DAE processing is stopped, dialog processing occurs, and DAE processing is restarted on the systems involved. There are some cases where a particular system may end up using different DAE parameters from those it was previously using. The following discussion illustrates possible results.

For this discussion there are two systems (SY1 and SY2), and five ADYSETxx members involved. Members ending with G1 and G2 include GLOBAL(DSN,OPTIONS) parameters. Members ending with S1 and S2 have SHARE(DSN,OPTIONS) without GLOBAL options. The center column in the table below indicates the system where the TSO user is issuing the TAKEDUMP request.

Start State		TSO	Final State	
SY1	SY2		SY1	SY2
G1	G1	SY1	G1	G1[1]
S1	S2	SY1	S1	S1[2]
S1	S2	SY2	S2	S2[2]
S1	G1	SY1	S1	S1[2]
S1	G2	SY2	G2	G2[2]
G2	S2	SY2	S2	S2[2]
00	*	SY1	00	*[3]

Notes:

1. GLOBAL(DSN) systems remain synchronized.
2. When all systems are NOT in GLOBAL(DSN) mode, the system will be started using the member last active on the system where the IPCS TAKEDUMP dialog runs.
3. No change to other systems if all work is done on a system which is not sharing the DAE data set. Here, ADYSET00 contains the default IBM—supplied values.

The system will generate the next dump for the symptom string. After that dump, DAE resumes suppressing dumps for the symptom string.

Note: Despite specifying action code T, the dump might still be suppressed. See "Determining Why a Dump Was Suppressed" on page 17-12 for the reasons, other than DAE suppression.

Editing the DAE Data Set

Edit the DAE data set, using Interactive System Productivity Facility (ISPF) edit. For ISPF edit, see *z/OS ISPF Dialog Developer's Guide and Reference*. You must have WRITE access to the DAE data set. Once in ISPF Edit, use the Edit macro ADYUPDAT as described below.

In the edit session, type one of the following on the command line, place the cursor on the symptom string line for the dump, and press ENTER. If the cursor is on the command line, the first symptom string is used. Note that DAE must be stopped before these actions and started again after.

```
ADYUPDAT TAKEDUMP
ADYUPDAT NODUMP
```

ADYUPDAT TAKEDUMP requests that the next dump be generated for this symptom string. SLIP can still suppress the dump.

ADYUPDAT NODUMP undoes the effect of TAKEDUMP, if it was in effect. Otherwise, NODUMP results in no action.

In the edit session, you can also delete every symptom string that has not been updated within a specified number of days. You must SAVE the DAE data set for the deletions to take effect. To request the deletions, enter on the command line:

```
ADYUPDAT CLEANUP nnn
```

Where:

nnn number of days a record has not been updated for it to be selected for deletion. The default is 180 days.

ADYUPDAT always issues a status message reflecting the outcome of the command.

Stopping, Starting, and Changing DAE

If an ADYSET00 parmlib member is used and the DAE data set allocated, DAE starts during IPL. Normally, DAE should run all the time that the system is running.

An operator can stop and start DAE with the following steps. One reason to do these steps would be to change to a different ADYSETxx parmlib member with different parameters.

Stopping DAE

The operator can stop DAE with a SET DAE command that specifies the ADYSET01 parmlib member, which contains a DAE=STOP statement:

```
SET DAE=01
```

Starting DAE

The operator can start DAE with a SET DAE command that specifies an ADYSETxx parmlib member that contains the DAE=START parameter, such as an installation-provided ADYSET03 parmlib member:

```
SET DAE=03
```

Changing DAE Processing in a Sysplex

The operator can change all DAE processing in a sysplex, if desired. For example, the operator can do the following to make all systems in a sysplex use a different ADYSETxx member:

1. Stop DAE processing using the IBM-supplied ADYSET01 member:

   ```
   ROUTE *ALL,SET DAE=01
   ```

   Another way to stop DAE processing on all systems in a sysplex is to specify in the SET DAE command an ADYSETxx member containing a GLOBALSTOP parameter.
2. Start DAE processing using, for example, the ADYSET04 member:

   ```
   ROUTE *ALL,SET DAE=04
   ```

Using a SLIP Command to Suppress Dumps

Some dumps are almost never needed. For example, some abend codes tell the diagnostician enough to solve the problem. For these codes, place SLIP operator commands in an IEASLPxx parmlib member to suppress the unneeded dumps. The IBM-supplied IEASLP00 member contains the SLIP commands to suppress abend dumps that are seldom needed.

Using SLIP to suppress dumps also suppresses message IEA995I, which contains symptom dump information. The system may document the abend in a LOGREC error record.

To suppress dumps for an abend code, specify a SLIP operator command with one of the following ACTION parameters. Place all the SLIP commands in an IEASLPxx parmlib member and activate the IEASLPxx member with the following command in a COMMNDxx or IEACMDxx member that is always used:

```
CMD='SET SLIP=xx'
```

SLIP Parameter	Dumps Suppressed
ACTION=NODUMP	All dumps
ACTION=NOSVCD	SVC dumps
ACTION=NOSYSA	ABEND SYSABEND dumps
ACTION=NOSYSM	ABEND SYSMDUMP dumps
ACTION=NOSYSU	ABEND SYSUDUMP dumps

For example, to suppress SVC dumps and ABEND SYSMDUMP dumps for abend code X'B37', add the following to IEASLPxx:

```
SLIP SET,COMP=B37,ACTION=(NOSVCD,NOSYSM),END
```

References
- See *z/OS MVS Initialization and Tuning Reference* for the IEASLPxx member.
- See *z/OS MVS System Commands* for the SLIP operator command.

Using an ABEND Macro to Suppress Dumps

A program can suppress a dump by issuing an ABEND macro without a DUMP parameter. Application programmers should not specify a DUMP parameter when a symptom dump can provide enough information for diagnosis.

Reference

See *z/OS MVS Programming: Assembler Services Reference ABE-HSP* for the ABEND macro.

Using Installation Exit Routines to Suppress Dumps

An installation can add installation exit routines to suppress dumps. Use IEAVTABX if you want to suppress abend dumps based on the job name, abend code, or other information in the system diagnostic work area (SDWA). Use IEAVTSEL if you want to discard an SVC or SYSMDUMP dump based on information in the dump header or from DAE. Use JES2 exit 4 or JES3 exit IATUX34 to suppress different types of dumps.

Exit	Processing	Dump Suppression
IEAVTABX	Before any ABEND dump	Routine(s) can place a return code of 8 in register 15 to suppress the requested dump.
IEAVTSEL	After an SVC dump or ABEND SYSMDUMP dump, if the dump was not suppressed by DAE	Routine(s) can clear the dump data set.
JES2 exit 4 or JES3 IATUX34	For any JCL statement	Can change the DSNAME parameter on a dump DD statement to DUMMY to suppress the dump.

References
- See *z/OS MVS Installation Exits* for IEAVTABX and IEAVTSEL.
- See *z/OS JES2 Installation Exits* for the JES2 exit 4 routine.
- See *z/OS JES3 Customization* for the JES3 IATUX34 exit routine.

Determining Why a Dump Was Suppressed

If an intended dump is missing, use this list to decide why. The list gives reasons why dumps are suppressed, including the ways discussed in this chapter. In planning for problem determination, be aware of all of these ways so that your installation does not suppress intended dumps.

- **DAE suppression of dumps.** See “Using DAE to Suppress Dumps” on page 17-1.
- **SLIP command that suppresses all dumps for an abend code.** See “Using a SLIP Command to Suppress Dumps” on page 17-11.

- **An ABEND macro without a DUMP parameter.** See “Using an ABEND Macro to Suppress Dumps” on page 17-12.
- **An MVS installation exit routine that suppresses the dump.** See “Using Installation Exit Routines to Suppress Dumps” on page 17-12.
- **Resource Access Control Facility (RACF) control of programs in an address space to be dumped**: Beginning with RACF 1.8.1, the installation can protect ABEND dumps of programs using the FACILITY class. The protection can keep you from accessing a dump.
- **Dump on another system blocked by SYSDCOND in the PROBDESC area and the IEASDUMP.QUERY routine**. A dump on another system in a sysplex is requested by a DUMP command or SDUMPX macro with a REMOTE parameter. If the area specified by the PROBDESC parameter contains SYSDCOND, the dump on the other system is not written because of either of the following on the other system:
  – No IEASDUMP.QUERY routine exists
  – No IEASDUMP.QUERY routine returns a code of 0
- **Dump suppressed by CHNGDUMP command**. If a CHNGDUMP command specifies NODUMP for SVC dumps:
  – All SVC dumps on the system are suppressed.
  – If a DUMP command or SDUMPX macro includes a REMOTE parameter, the dump on the local system and the dumps on other systems in the sysplex are suppressed.
- **Dump on another system suppressed by CHNGDUMP command on the other system**. If a DUMP command or SDUMPX macro includes a REMOTE parameter and a CHNGDUMP command previously entered on another system in the sysplex specifies NODUMP for SVC dumps, the SVC dump on the other system is suppressed. The dump on the local system is written.

The system can also place the dump in another data set, so that it is not in the original data set specified in a message you received:

- **An installation exit routine at JES2 exit 4 or at JES3 exit IATUX34 can change the dump data set name.**
- **DUMPDS operator command can redirect SVC dump output.** The command can redirect SVC dump output to other SYS1.DUMPxx data sets.

References

- See *z/OS Security Server RACF Security Administrator’s Guide* for the FACILITY class to control access to program dumps.
- See *z/OS MVS System Commands* for the DUMP, DUMPDS, and SLIP commands.
- See *z/OS MVS Installation Exits* for the IEAVTABX and IEAVTSEL exit routines.
- See *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU* for the SDUMPX macro.
- See *z/OS MVS Programming: Authorized Assembler Services Guide* for the IEASDUMP.QUERY routine.
- See *z/OS JES2 Installation Exits* for the JES2 exit 4 routine.
- See *z/OS JES3 Customization* for the JES3 IATUX34 exit routine.

Chapter 18. Messages

You don't want to see system messages, you just want everything to work right. But since you have to use them, here's how.

The system issues messages to do the following:

- Tell the operator or system programmer of progress and problems in system processing
- Ask the operator to take actions and make decisions
- Tell the application programmer how the system ran the application program and of problems in the application program

The system issues messages from the base control program components and a variety of subsystems, products, and applications. Applications running under the system can also issue their own messages.

Major Topics

This chapter contains the following topics about messages:

- “Producing Messages”
- “Receiving Messages” on page 18-2
- “Planning Message Processing for Diagnosis” on page 18-4

Producing Messages

You can get the system to produce a message by issuing a macro in any program or by asking an operator to enter a command. The macros and command are:

- LOG operator command to write a message to the SYSLOG
- WTL macro to write a message to the SYSLOG
- WTO macro to write a message to the operator
- WTOR macro to write a message to the operator and request a reply

Use the following for related activities:

- DOM macro to delete an operator message or group of messages from the display screen of a console
- REPLY operator command to answer a message
- WRITELOG operator command to start, stop, or print the SYSLOG and to change the output class for the SYSLOG

References

- See *z/OS MVS Programming: Assembler Services Reference ABE-HSP* for DOM and *z/OS MVS Programming: Assembler Services Reference IAR-XCT* for the WTO, WTOR, amd WTL macros.
- See *z/OS MVS System Commands* for the LOG, REPLY, and WRITELOG commands.

Receiving Messages

The system issues messages through WTO and WTOR macros to the following locations. Routing codes determine where the messages are displayed or printed.
- Console
- Extended console
- Hard-copy log
- Job log
- SYSOUT data set

The system issues messages through the WTL macro to the SYSLOG.

The access methods issue messages directly to one of the following locations:
- Display terminal
- Output data set

Console

Messages sent to a console with master authority are intended for the operators. The system writes in the hard-copy log all messages sent to a console, regardless of whether the message is displayed.

Hard-Copy Log

The hard-copy log is a record of all system message traffic:
- Messages to and from all consoles
- Commands and replies entered by the operator

In a dump, these messages appear in the master trace. With JES3, the hard-copy log is always written to the SYSLOG. With JES2, the hard-copy log is usually written to the SYSLOG but can be written to a console printer, if the installation chooses.

System Log

The SYSLOG is a SYSOUT data set provided by the job entry subsystem (either JES2 or JES3). SYSOUT data sets are output spool data sets on direct access storage devices (DASD). An installation should print the SYSLOG periodically to check for problems. The SYSLOG consists of the following:
- All messages issued through WTL macros
- All messages entered by LOG operator commands
- Usually, the hard-copy log
- Any messages routed to the SYSLOG from any system component or program

Job Log

Messages sent to the job log are intended for the programmer who submitted a job. Specify the system output class for the job log in the MSGCLASS parameter of the JCL JOB statement.

SYSOUT Data Set

Messages sent to a SYSOUT data set are intended for a programmer. These messages are issued by an assembler or compiler, the linkage editor and loader, and an application program.

To make all messages about a program appear in the same SYSOUT listing, specify the same class for the SYSOUT data set and in the MSGCLASS parameter on the JCL JOB statement.

Receiving Symptom Dumps

A symptom dump is a system message, either message IEA995I or a numberless message, which provides some basic diagnostic information for diagnosing an abend. Often the symptom dump information can provide enough information to diagnose a problem.

Example: Symptom Dump Output

The following example shows the symptom dump for an abend X'0C4' with reason code X'4'. This symptom dump shows that:
- Active load module ABENDER is located at address X'00006FD8'.
- The failing instruction was at offset X'12' in load module ABENDER.
- The address space identifier (ASID) for the failing task was X'000C'.

```
IEA995I SYMPTOM DUMP OUTPUT
SYSTEM COMPLETION CODE=0C4  REASON CODE=00000004
 TIME=16.44.42  SEQ=00057  CPU=0000  ASID=000C
 PSW AT TIME OF ERROR  078D0000   00006FEA  ILC 4  INTC 04
   ACTIVE LOAD MODULE=ABENDER   ADDRESS=00006FD8  OFFSET=00000012
   DATA AT PSW  00006FE4 - 00105020  30381FFF  58E0D00C
   GPR  0-3  FD000008  00005FF8  00000014  00FD6A40
   GPR  4-7  00AEC980  00AFF030  00AC4FF8  FD000000
   GPR  8-11 00AFF1B0  80AD2050  00000000  00AFF030
   GPR 12-15 40006FDE  00005FB0  80FD6A90  00006FD8
 END OF SYMPTOM DUMP
```

Symptom dumps appear in the following places:
- For SYSUDUMP and SYSABEND ABEND dumps: in message IEA995I, which is routed to the job log.
- For a SYSMDUMP ABEND dump: in message IEA995I in the job log and in the dump header record.
- For an SVC dump: in the dump header record.
- For any dump in a Time Sharing Option/Extensions (TSO/E) environment: displayed on the terminal when requested by the TSO/E PROFILE command with the WTPMSG option.
- In response to a DISPLAY DUMP,ERRDATA operator command, which is used to display information from SYS1.DUMPxx data sets on direct access.

If the information in a symptom dump is enough for diagnosis, do not provide a DD statement for a dump.

References
- See *z/OS MVS Routing and Descriptor Codes* for the routing codes for messages issued by the operating system.
- See Chapter 9, “Master Trace” on page 9-1 for information on master trace.
- See *z/OS MVS JCL Reference* for the JOB statement.
- See Chapter 5, “ABEND Dumps” on page 5-1 for information about the ABEND dump header record
- See Chapter 2, “SVC Dump” on page 2-1 for information about the SVC dump header record
- See *z/OS TSO/E Command Reference* for the PROFILE command.

Planning Message Processing for Diagnosis

Your installation can change message processing in a number of ways to optimize diagnosis.

Your installation can:
- Control message location
- Suppress messages
- Automate message processing
- Not retain action messages
- Suppress the symptom dump message (IEA995I)

This section can help you find the information you need to optimize message processing for your installation.

Controlling Message Location

An installation can change the following:
- The routing codes for specific messages to control if a message is displayed on a console
- On which console a message is displayed.

The routing codes are specified or changed by the following:
- A WTO or WTOR macro specifies the routing code for the message that the macro creates.
- A WTO/WTOR installation exit routine can change the routing code for any WTO or WTOR message. This exit routine is the routine named in the USEREXIT parameter in the MPFLSTxx parmlib member.
- The JES3 MSGROUTE initialization statement can change the routing code of JES3 console messages.
- The JES3 CONSOLE initialization statement can control which messages a JES3 console receives.
- Subsystem interface listeners can change the routing codes.

Suppressing Messages

An installation can use the following to suppress messages. Suppressed messages do not appear on a console.
- An MPFLSTxx parmlib member can specify that a message is to be suppressed. A suppressed message is not displayed on any console, but is written only to the hard-copy log.
- A WTO/WTOR installation exit routine can suppress any WTO/WTOR messages or override suppression. This exit routine is the routine named in the USEREXIT parameter in the MPFLSTxx parmlib member.
- The JES2 installation exit routine, Exit 10, can suppress messages issued by the JES2 main task.
- The JES3 installation exit routine, IATUX31, can suppress messages routed to JES3 consoles.
- The CONTROL V operator command can suppress messages by specifying the message levels to be displayed at a console.
- VARY operator command can change the messages received by a console by specifying the routing codes of messages to be displayed.
- The LEVEL keyword on the CONSOLE statement in the CONSOLxx parmlib member also can suppress messages by specifying the message levels to be

displayed at a console. The ROUTCODE keyword can specify the routing codes of messages to be displayed at the console.

Automating Message Processing

An MPFLSTxx parmlib member can specify that a message is to be passed to an automation subsystem, such as NetView. The automation subsystem can perform actions that the operator would have performed without operator intervention.

Not Retaining Action Messages

The MPFLSTxx parmlib member can specify that a message is not to be retained by the action message retention facility (AMRF). An operator cannot recall to the screen action messages that are not retained.

Suppressing Symptom Dumps (IEA995I)

The following table lists ways to suppress symptom dumps for ABEND dumps. These ways suppress only message IEA995I; symptom dumps continue to appear in other locations.

Dump Type	CHNGDUMP Operator Command used to Suppress Symptom Dump	Parmlib Member used to Suppress Symptom Dump
SYSABEND ABEND	CHNGDUMP SET,SYSABEND,SDATA=(NOSYM)	IEAABD00
SYSMDUMP ABEND	CHNGDUMP SET,SYSMDUMP=(NOSYM)	IEADMR00
SYSUDUMP ABEND	CHNGDUMP SET,SYSUDUMP,SDATA=(NOSYM)	IEADMP00

References

- See *z/OS MVS Planning: Operations* for controlling message display, suppressing messages, AMRF, and automating messages in a sysplex.
- See *z/OS MVS Routing and Descriptor Codes* for routing codes.
- See *z/OS MVS Programming: Assembler Services Reference IAR-XCT* for the WTO and WTOR macros.
- See *z/OS MVS Initialization and Tuning Reference* for the MPFLSTxx and CONSOLxx members.
- See *z/OS MVS Installation Exits* for the exit routine named in the USEREXIT parameter and for IEAVMXIT.
- See *z/OS JES3 Initialization and Tuning Reference* for the JES3 initialization statements.
- See *z/OS JES2 Installation Exits* for Exit 10.
- See *z/OS JES3 Customization* for the IATUX31 exit.
- See *z/OS MVS System Commands* for the CONTROL V and the CHNGDUMP operator command.

Appendix. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Programming Interfaces Information

This manual primarily documents information that is NOT intended to be used as Programming Interfaces of z/OS.

This manual also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Programming Interface information

End of Programming Interface information

Trademarks

The following terms are trademarks of the IBM Corporation in the United States and/or other countries:

- CICS
- CT
- Current
- DB2
- DFSMS/MVS
- DFSMSdss
- DFSMSdfp
- DFSMShsm
- Enterprise System/9000
- FFST
- First Failure Support Technology
- Hiperspace
- IBM
- IBMLink
- IMS
- MVS
- MVS/ESA
- NetView
- OS/390
- RACF
- Resource Link
- RETAIN
- RMF
- S/390
- SOMobjects
- SP
- Sysplex Timer
- Tivoli
- VTAM
- z/Architecture
- z/OS
- z/OS.e
- z/VM
- zSeries
- 3090
- 400

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET is a registered trademark of SET Secure Electronic Transaction LLC.

Other company, product and service names may be the trademarks or service marks of others.

Index

Special characters

* control statement
  in SPZAP 16-22

A

a dump data set
  to receive Transaction dump 3-6
AB= parameter
  in GTF 10-6
ABDUMP= parameter
  in GTF 10-6
abend analysis
  obtain abend code 5-21
  obtain reason code 5-21
abend code
  in STATUS FAILDATA report 2-40, 3-18
ABEND dump
  analysis 5-20
  contents 5-8
  customize 5-14
  displaying options 5-9
  reasons for selection 1-3
  summary dump contents 5-13
  SYSABEND dump analysis 5-20
  SYSUDUMP dump analysis 5-20
abend dumps
  synopsis 5-1
ABEND dumps
  obtaining 5-3
ABEND macro
  for requesting ABEND dumps 5-6
  in dump customization 5-15, 5-16, 5-17, 5-19
  to suppress dumps 17-12
abnormal end
  indicated in logrec error record 14-1
ABSDUMP/ABSDUMPT control statement
  example 16-11
  in SPZAP 16-11, 16-18, 16-19
  parameter 16-18, 16-19
accessibility A-1
ACR (alternate CPU recovery)
  problem data 7-5
  system trace event 8-6
ACR trace entry
  in system trace 8-6
ADATA= parameter
  LISTLOAD control statement 15-3
address
  commonly bad 7-2
address space
  buffers for component trace 11-9
addressing mode
  system trace event 8-11
Advanced Program-to-Program Communication
  See APPC/MVS 11-25
ALIB parameter
  of AMDSADMP macro 4-15
ALL parameter
  dump option 5-9
ALLNUC parameter
  dump option 2-22, 3-8, 5-9
allocation
  automatically of dump data set 2-2, 3-3
  component trace 11-49
  pre-allocation of dump data set 2-7, 3-2
ALLPA parameter
  dump option 5-9, 6-5
ALLPDATA parameter
  dump option 5-9
ALLPSA parameter
  dump option 2-22, 3-8
ALLSDATA parameter
  dump option 5-9
ALLVNUC parameter
  dump option 5-9, 6-5
ALTR trace entry
  in system trace 8-7
AMBLIST output
  obtain 15-1
AMBLIST service aid
  control statement
    rules for coding 15-2
  description 15-1
  functions 15-6
  JCL statement 15-2
  LISTIDR control statement 15-5
  LISTLOAD control statement 15-3
  LISTLPA control statement 15-5
  LISTOBJ control statement 15-4
  mapping CSECTs in a load module or program object 15-8
  mapping the contents of the nucleus 15-13
  mapping the modules in the link pack area 15-12
  output 15-13
  reasons for selection 1-5
  tracing modifications to the executable code in a CSECT 15-11
AMDSADDD utility 4-23
  catalog requirement 4-25
  characteristics 4-23
  CLEAR option 4-24
  DEFINE option 4-24
  invocation 4-23
  REALLOC option 4-24
  space requirement 4-25
  syntax 4-24, 4-25
  unit requirement 4-25
  vollist requirement 4-25
  volser requirement 4-25
AMDSADMP macro 4-11
  assembly 4-31
  DUMP keyword 4-19
  example 4-16

AMDSADMP macro *(continued)*
  format for high-speed dump 4-11
  multiple versions, assembling 4-32
  parameter
    ALIB= 4-15
    COMPACT= 4-14
    CONSOLE= 4-12
    DDSPROMPT= 4-15
    DUMP 4-18
    DUMP= 4-14
    IPL= 4-11
    LNKLIB= 4-15
    MINASID 4-14
    MODLIB= 4-15
    MSG= 4-14
    NUCLIB= 4-15
    OUTPUT= 4-13
    PROMPT 4-14, 4-18
    REUSEDS= 4-14
    SYSUT= 4-12
    ULABEL= 4-12
    VOLSER= 4-12
  sample JCL 4-32
  stage-two generation 4-31
  symbol 4-11, 4-32
  syntax
    for high-speed dump 4-11
  SYS1.MACLIB data set
    assembly 4-32
AMRF (active message retention facility)
  for message retention 18-5
ANR record 14-8
APPC/MVS (Advanced Program-to-Program Communications/MVS)
  component trace 11-25
ASIDP trace option
  in GTF 10-18
    prompting 10-25
asynchronous Transaction dump 3-17
authorized program
  request dump 2-12, 3-6
automation
  of messages 18-5

B

BAKR instruction
  system trace event 8-8
BALR instruction
  system trace event 8-8
BASE control statement
  example 16-21
  in SPZAP 16-9, 16-19, 16-20, 16-21
  parameter 16-20
BASR instruction
  system trace event 8-8
BASSM instruction
  system trace event 8-8
BR trace entry
  in system trace 8-8
branch
  system trace event 8-8
branch instruction
  trace 8-2
BRANCH parameter
  to control summary dump 2-26
BSG trace entry
  in system trace 8-9
buffer
  for component trace 11-9
  logrec 14-18

C

CALL trace entry
  in system trace 8-15
CALLRTM macro
  for requesting ABEND dumps 5-7
  in dump customization 5-16, 5-17, 5-19
CANCEL operator command
  to request ABEND dump 5-7
catalog
  rebuild 16-17
CB parameter
  dump option 5-9, 6-5
CC-012 system control frame 4-37
CCHHR control statement
  in SPZAP 16-10, 16-21
  parameter 16-21
CCW trace option
  in GTF 10-19
CCW trace record
  formatted 10-41
  unformatted 10-82
CCWN parameter of GTF CCWP 10-25, 10-26
CCWP trace option
  DATA parameter 10-26
  in GTF 10-19, 10-25, 10-26
    I parameter 10-25
    IOSB parameter 10-26
    PCITAB parameter 10-26
    S parameter 10-25
    SI parameter 10-25
  prompting 10-25
central storage dump
  description 4-1
  of stand-alone dump 4-5
channel program data
  record 10-19
CHECKSUM control statement
  in SPZAP 16-22
  parameter 16-22
CHNGDUMP operator command
  in dump customization 2-30, 3-12, 5-17, 5-18, 5-20
  to change dump options 2-16, 5-9
  to suppress symptom dump 18-5
clear
  SVC dump 2-20
  Transaction dump 3-6
CLKC trace entry
  in system trace 8-15

combination of AMBLIST control statement 15-2
combination of GTF trace options 10-23
common storage tracking
  reasons for selection 1-5
COMPACT parameter
  of AMDSADMP macro 4-14
component trace
  description 11-1
  options
    for OPS 11-94
  reasons for selection 1-4
  start the external writer 11-16
  start the trace 11-12, 11-16, 11-19
  stop the external writer 11-17
  stop the trace 11-13, 11-17, 11-20
  sublevel 11-1, 11-3
  sublevel trace
    verifying 11-21
  SYSAPPC for APPC/MVS component 11-25
  SYSDLF for data lookaside facility (DLF) 11-39
  SYSDSOM for distributed SOM 11-41
  SYSGRS for global resource serialization 11-43
  SYSIEFAL for allocation component 11-49
  SYSIOS for IOS component trace 11-54
  SYSJES for JES common coupling services 11-60
  SYSjes2 for JES2 subsystem 11-71
  SYSLLA for LLA 11-73
  SYSLOGR for system logger 11-74
  SYSOMVS for z/OS UNIX 11-81
  SYSOPS for OPS component 11-92
  sysplex 11-18
  SYSRRS for RRS component 11-97
  SYSRSM for RSM 11-105
  SYSSPI for SPI component 11-120
  SYSVLF for VLF component 11-121
  SYSWLM for WLM component 11-125
  SYSXCF for XCF component 11-127
  SYSXES for cross-system extended services 11-131
  verifying 11-21
  viewing trace data 11-23
compression
  use for dumps 2-9
CONSOLE control statement
  example 16-23
  in SPZAP 16-22, 16-23
console supported by stand-alone dump 4-12
CONSOLE= parameter
  of AMDSADMP macro 4-12
  specifying the SYSC constant 4-12
consolidate
  data from GTF traces 10-32
control records
  GTF trace record
    unformatted 10-77
control registers
  format in a dump 2-47, 3-25
control statement
  for AMBLIST service aid 15-2
  for SPZAP 16-17
copy
  SVC dump 2-20
  Transaction dump 3-6
copy a dump
  DASD to DASD 4-45
  multiple devices to DASD 4-46
  tape to DASD 4-45
COPYDUMP subcommand
  to obtain component trace data 11-23
COPYTRC subcommand
  for component traces 11-23
COUPLE parameter
  dump option 2-22, 3-8
cross memory instruction
  system trace event 8-9
cross memory processing mode
  problem data 7-4
cross-system extended services
  component trace 11-131
CRW record 14-8
CSA dumped by stand-alone dump 4-5
CSA parameter
  dump option 5-9
CSCH trace entry
  in system trace 8-13
CSCH trace option
  in GTF 10-19
CSCH trace record
  formatted 10-42
CSECT
  tracing modifications to the executable code in a CSECT 15-11
CSECT identification record (IDR)
  print 15-47
CSECT name
  NAME control statement
    of SPZAP 16-2
CSECTs in a load module or program object
  mapping with AMBLIST service aid 15-8
CTIIEFxx parmlib member
  for SYSIEFAL component trace 11-49
CTnAPPxx parmlib member
  for SYSAPPC component trace 11-26
CTnBPXxx parmlib member
  for SYSOMVS component trace 11-82
CTnGRSxx parmlib member
  for SYSGRS component trace 11-44
CTnIOSxx parmlib member
  for SYSIOS component trace 11-55
CTnJESON parmlib member
  for SYSJES component trace 11-62
CTnJESxx parmlib member
  for SYSJES component trace 11-62
CTnLOGxx parmlib member
  for SYSLOGR component trace 11-77
CTnOPSxx parmlib member
  for SYSOPS component trace 11-93
CTnRRSxx parmlib member
  for SYSRRS component trace 11-98
CTnRSMxx parmlib member
  for SYSRSM component trace 11-106

CTnXCFxx parmlib member
for SYSXCF component trace 11-128
CTnXESxx parmlib member
for SYSXES component trace 11-133
CTRACE
See component trace 11-1
CTRACE subcommand
for SYSAPPC component trace 11-29
for SYSOMVS component trace 11-84
for SYSOPS component trace 11-95
for SYSWLM component trace 11-126
for SYSXCF component trace 11-130
for SYSXES component trace 11-135
FULL report 11-96
SHORT report 11-96
customization
of nucleus area in dump 2-30, 3-11, 5-15
customize
master trace 9-2

D

DAE (dump analysis and elimination)
description 17-1
DAE data set
editing 17-8
managing 17-8
removing old symptom strings 17-8
DAE Display panel
invoking 17-8
DAE service aid
reasons for selection 1-5
DASD (direct access storage device) 4-1, 4-6, 4-11, 4-12, 4-29, 4-33
allocation 4-22
AMDSADDD utility 4-23
data set 4-22
types 4-22
DASD-SIM recovery record 14-8
data
for diagnosis 7-1
inspect with SPZAP 16-2
modify with SPZAP 16-2
data inspection
use SPZAP 16-2
data lookaside facility
See DLF 11-39
data modification
use SPZAP 16-2, 16-3, 16-4, 16-5, 16-11, 16-21
DATA parameter of GTF CCWP 10-25, 10-26
data set
automatically allocated for dumps 2-2, 3-3
to receive ABEND dump 5-3
data space
buffers for component trace 11-9
DCB parameter
for component trace data set 11-15
DD statement
in AMBLIST service aid
anyname 15-2
SYSIN data set 15-2
DD statement *(continued)*
in AMBLIST service aid *(continued)*
SYSPRINT data set 15-2
in IFCDIP00 14-4
in logon procedure 5-6
in SPZAP
SYSABEND data set 16-17
SYSIN data set 16-4, 16-17, 16-21, 16-23
SYSLIB data set 16-2, 16-3, 16-4, 16-5, 16-11, 16-15, 16-16, 16-21, 16-23
SYSPRINT data set 16-11, 16-16
in stand-alone dump 4-31, 4-45
SYSIN data set 4-30
SYSPRINT data set 4-30
SYSPUNCH data set 4-32
SYSLIB data set 4-31
SYSMDUMP 5-5
SYSOUT 5-5, 6-2
SYSUT2 statement 4-45
DDN= parameter
LISTIDR control statement 15-5
LISTLOAD control statement 15-3
LISTOBJ control statement 15-4
DDR record 14-8
DDSPROMPT parameter
of AMDSADMP macro 4-15
DEBUG= parameter
in GTF 10-7
DEFAULTS parameter
dump option 2-22, 3-8
detail edit report
produced by
EREP 14-20
IPCS VERBEXIT LOGDATA subcommand 14-20
software record 14-20
symptom record 14-26
DIAGxx parmlib member
for requesting GFS trace 13-1
disability A-1
disabled
summary dump 2-26
disk initialization program - IFCDIP00 14-2
dispatch
system trace event 8-12
DISPLAY operator command
for SYS1.DUMPxx information 18-3
to display dump options 2-16, 5-9
to find SVC dump 2-17
DIV (data-in-virtual) 11-109
DLF (data lookaside facility)
component trace 11-39
DM parameter
dump option 5-9, 6-5
documents, licensed xviii
DSOM (distributed SOMobjects)
component trace 11-41
DSP trace entry
in system trace 8-12
DSP trace option
in GTF 10-19

DSP trace record
  comprehensive
    unformatted 10-83
  formatted 10-44
  minimal
    unformatted 10-83
dump
  automatically allocated data set 2-2, 3-3
  customization 5-15
  customize contents of ABEND dump 5-14
  displaying options 2-16, 5-9
  dump analysis
    summary SVC 2-32
    summary Transaction 3-13
    SVC 2-32
    Transaction 3-13
  find 2-17
  notification when rapidly recurring 17-4
  pre-allocated data set 2-7, 3-2
  reasons dumps are suppressed 17-12
  requesting a suppressed dump 17-9
  suppress by abend code 17-11
  suppress by ABEND macro 17-12
  suppress by DAE 17-1
  suppress by installation exit routines 17-12
  suppress by RACF 17-12
  suppress by SLIP trap 17-11
  suppressing when rapidly recurring 17-3
  suppression 17-1
  time 2-17
  title in SVC dump 2-17
dump analysis and elimination
  See DAE 17-1
DUMP command
  SLIP command 11-11
  to obtain component trace 11-11
dump data sets
  control 3-3
  to receive Transaction dump
    specify 3-3
dump grab bag 7-1
  storage overlay 7-1
DUMP keyword 4-19
DUMP operator command
  in dump customization 2-31
  to request SVC dump 2-12
DUMP parameter
  of AMDSADMP macro 4-14, 4-18
dump selection
  ABEND dump 1-1
  SNAP dump 1-1
  stand-alone dump 1-1
  SVC dump 1-1
dump tailor option
  for stand-alone dump 4-19
dump title 2-38, 3-15
  in AMBLIST service aid
    LISTIDR control statement 15-5
    LISTLOAD control statement 15-3
    LISTOBJ control statement 15-4
  specification 15-3, 15-4, 15-5
DUMP/DUMPT control statement
  in SPZAP 16-3, 16-4, 16-5, 16-6, 16-21, 16-23
  parameter 16-23
DUMP/DUMPT control statement example
  explanation of second control statements 16-5
  inspecting and modifying two CSECTs 16-4
  modifying a CSECT in a load module 16-3
  using SPZAP to modify a CSECT 16-6
DUMPDS command
  make data set available 2-15
DUMPDS operator command
  to clear SYS1.DUMPxx data set 2-20
DUMPOPT or DUMPOPX parameter
  in dump customization 5-16, 5-17, 5-19
dumps
  description 1-3
dvolser 4-49
dynamic invocation
  of SPZAP 16-28

E

EID (event identifier) for GTF 10-75
EMS trace entry
  in system trace 8-15
enabled
  summary dump 2-26
ENQ parameter
  dump option 5-9
entry
  system trace 8-3
environmental data
  definition 14-6
  record header information 14-7
environmental record 14-6
EOD record 14-8
  RDE option 14-8
ERR parameter
  dump option 5-9, 6-5
error
  record in logrec 14-1
error identifier
  in STATUS WORKSHEET report 2-39, 3-17
  in VERBEXIT LOGDATA report 2-46, 3-24
error statistic
  definition 14-6
ESTAE or ESTAEX macro
  in dump customization 5-16, 5-18, 5-19
ESTAI parameter
  in dump customization 5-16, 5-18, 5-19
ETR recovery record 14-8
exit 4
  for JES2
    to suppress dumps 17-12
exit routine
  to suppress dumps 17-12
EXT trace entry
  in system trace 8-15
EXT trace option
  in GTF 10-19

EXT trace record
    comprehensive
        unformatted 10-84
    formatted 10-47
    minimal
        unformatted 10-84
extended sequential data set
    to hold large dumps 2-7
external interruption
    record 10-19
external writer
    source JCL 11-14
extract
    GTF trace data from dumps 10-32

F

failing instruction 2-42, 3-19
FID (format identifier) 10-76
FID (format identifier) for GTF 10-76
fixed link pack area
    map 15-5
FLIH (first level interrupt handler)
    problem data
        saved by external FLIH 4-63
        saved by I/O FLIH 4-62
        saved by SVC FLIH 4-61
FLPA parameter
    of LISTLPA control statement 15-5
FORCE ARM command 10-17
format of the logrec buffer 14-19
formatting
    master trace 9-4
formatting the Logrec buffer 14-18
FRR (functional recovery routine)
    in dump customization 5-16, 5-18, 5-19
FRR (functional recovery routine) data
    record 10-21
FRR trace record
    formatted 10-49

G

general purpose registers
    format in a dump 2-47, 3-25
GFS trace
    requesting 13-1
global resource serialization
    component trace 11-43
goff listing
    AMBLIST output for LISTOBJ with GOFF records. 15-20
GRSQ parameter
    dump option 2-22, 3-8, 5-9
GTF (generalized trace facility)
    combining trace options 10-23
    prompting 10-23
    prompting keywords in SYS1.PARMLIB 10-29
    specification of a system event 10-30
    specification of GTF trace for a system event 10-29
    starting 10-13
GTF (generalized trace facility) *(continued)*
    starting GTF 10-15
        prompting 10-23
        START command 10-13
        with internal tracking mode 10-14
    starting with data recorded on a device 10-18
    STOP command 10-16, 10-18
    storing trace options in SYS1.PARMLIB 10-14
    trace VTAM remote network activity 10-16
GTF (generalized trace facility) trace
    cataloged procedure 10-3
    CCW trace record
        formatted 10-41
    CCW trace records
        unformatted 10-82
    control record 10-77
    CSCH trace record
        formatted 10-42
    customization 10-4
    definition of trace options 10-4
    DSP comprehensive trace record
        unformatted 10-83
    DSP minimal trace record
        unformatted 10-83
    DSP trace record
        formatted 10-44
    EID (event identifier) for USR trace records 10-75
    EXT comprehensive trace record
        unformatted 10-84
    EXT minimal trace record 10-84
    EXT trace record
        formatted 10-47
    FID (format identifier) for USR trace records 10-76
    formatted output 10-37
    FRR trace record
        formatted 10-49
    generation of trace record 10-2
    HEXFORMAT trace record 10-50
    HSCH trace record
        formatted 10-42
    I/O trace record
        unformatted 10-87
    IBM defaults 10-3
    IBM-supplied catalogued procedure 10-3
    in GTF 10-1
    IOX trace record
        formatted 10-51
    lost data record 10-79
        unformatted 10-80
    lost event record 10-40
    LSR trace record
        formatted 10-54
    merge with component traces 10-33
    MSCH trace record
        formatted 10-55
    output direction 10-1
    parmlib member options 10-3
    PGM trace record
        formatted 10-56
    PI comprehensive trace record
        unformatted 10-88

GTF (generalized trace facility) trace *(continued)*
PI minimal trace record
unformatted 10-88
PI trace record
formatted 10-56
receive 10-31
records 10-34
request reasons 10-1
RNIO trace record
formatted 10-57
RR comprehensive trace record
unformatted 10-88
RR minimal trace record
unformatted 10-89
RSCH trace record
formatted 10-58
SDSP trace record
formatted 10-44
setting up a catalogued procedure 10-4
SLIP DEBUG trace record
unformatted 10-92
SLIP trace record
formatted 10-59
unformatted 10-89
SLIP user trace record
unformatted 10-91
source index record 10-40
SRB trace record
formatted 10-64
SRM comprehensive trace record
unformatted 10-93
SRM minimal trace record
unformatted 10-93
SRM trace record
formatted 10-65
SSCH trace record
formatted 10-66
unformatted 10-93
STAE trace record
formatted 10-67
starting GTF
how to start 10-10, 10-11
START command 10-11
storage requirement determination 10-9
SUBSYS trace record
formatted 10-50
SVC comprehensive trace record
unformatted 10-94
SVC minimal trace record
unformatted 10-95
SVC trace record
formatted 10-69
system data record
unformatted 10-81
system data records 10-82
SYSTEM trace record
formatted 10-50
time stamp record 10-39
trace an event in indexed VTOC processing 10-2
trace record format 10-18
unformatted output 10-77
GTF (generalized trace facility) trace *(continued)*
use of IPCS to print output 10-2
use with system trace 10-2
USR trace record
formatted 10-70
GTF (Generalized trace facility) trace
SLIP standard trace record
unformatted 10-91
GTF cataloged procedure
IBM defaults 10-3
GTF START command parameter
parm member
BLOK= 10-6
GTF trace
ASIDP option 10-18
CCWP option 10-19
CSCH option 10-19
DSP option 10-19
EXT option 10-19
HSCH option 10-19
IO option 10-19
IOX option 10-19
JOBNAMEP option 10-20
MSCH option 10-20
PCI option 10-20
PI option 10-21
PIP option 10-21
reasons for selection 1-4
RNIO option 10-21
SI option 10-21
SIOP option 10-21
SLIP option 10-21
SRM option 10-21
SSCH option 10-21
SSCHP option 10-21
SVC option 10-22
SVCP option 10-22
SYS option 10-22
SYSM option 10-22
SYSP option 10-22
TRC option 10-22
USR option 10-23
GTF trace event
record 10-22
GTF trace option
combining options 10-23
prompting for 10-23
USRP option 10-23
GTFTRACE subcommand
output from dump or data set 10-37
to format GTF trace 10-31

H

halt subchannel operation
GTF record 10-19
hardcopy log
and master trace 9-1
hardware
error 14-1

header record
  for incident record 14-7
  for logrec data set
    format 14-8
    used by EREP 14-7
    used by recording routine 14-7
HEXFORMAT trace record
  formatted 10-50
high speed dump 4-1
high speed dump program
  example 4-16
high-speed version
  of stand-alone dump 4-11
high-speed version of stand-alone dump 4-5
hiperspace data
  dump 2-21, 3-8, 5-8, 6-5
HSCH trace entry
  in system trace 8-13
HSCH trace option
  in GTF 10-19
HSCH trace record
  formatted 10-42

I

I/O interruption
  GTF record 10-19
I/O operation
  system trace event 8-13
I/O trace entry
  in system trace 8-15
I/O trace record
  unformatted 10-87
IATUX34 installation exit
  for JES3
    to suppress dumps 17-12
identifier
  for formatted GTF formatted trace record 10-36
  for system trace entries 8-4
IDRC (improved data recording capability) feature
  COMPACT parameter 4-14
IDRDATA control statement
  example 16-3, 16-4, 16-5, 16-6, 16-21
  in SPZAP 16-3, 16-4, 16-5, 16-6, 16-21, 16-24
  parameter 16-24
IEAABD00 parmlib member
  in dump customization 5-16
  in dump suppression 18-5
IEACMD00 parmlib member
  in dump customization 2-30, 3-12
IEADMP00 parmlib member
  in dump customization 5-19
  in dump suppression 18-5
IEADMR00 parmlib member
  in dump customization 5-17
  in dump suppression 18-5
IEASLPxx parmlib member
  for SLIP operator command 17-11
IEATDUMP macro
  to request Transaction dump 3-6
IEAVADFM exit
  in dump customization 5-17, 5-20, 6-4
IEAVADUS exit
  in dump customization 5-17, 5-20, 6-4
IEAVTABX exit
  in dump customization 5-17, 5-18, 5-20
IEAVTABX installation exit
  to suppress dumps 17-12
IEAVTSDT program 2-39, 3-17
IEAVTSEL installation exit
  to suppress dumps 17-12
IFCDIP00 - disk initialization program
  application 14-2
  changing logrec data set size 14-2
  reinitializing logrec data set 14-2
IFCDIP00 service aid
  function 14-2
  initializing logrec data set 14-2
  reallocate space on logrec data set 14-3
    JCL example 14-3
  reinitializing logrec data set 14-4
    JCL example 14-4
incident record
  on logrec data set
    content 14-7
    record header 14-7
    size 14-7
  on logrec log stream
    content 14-7
    size 14-7
initialization error messages
  in stand-alone dump 4-6
installation exit routine
  in dump customization 5-17, 5-18, 5-20, 6-4
  to suppress dumps 17-12
instruction address trace 4-5
  reasons for selection 1-4
interpretation of software record 14-19
interruption
  code 10-28
  I/O 10-27
  program 10-21, 10-28
  supervisor 10-22, 10-28
  SVC interrupt 10-22, 10-28
  system trace event 8-15
introduction
  SNAP dumps 6-1
IO parameter
  dump option 5-9, 6-5
IO trace option
  in GTF 10-19
IO=SSCH= keyword
  in GTF 10-28
  prompting 10-28
IOP trace option
  in GTF 10-27
  prompting 10-27
IOS
  component trace 11-54
IOS recovery record 14-8
IOSB parameter of GTF CCWP 10-25, 10-26

IOX trace option
in GTF 10-19
IOX trace record
formatted 10-51
IPCS service aid
reasons for selection 1-5
IPCS subcommand
VERBEXIT LOGDATA
to format in-storage logrec buffer 14-18
IPL record 14-8
RDE option 14-8
IPL/outage recorder
function 14-8
IPL= parameter
of AMDSADMP macro 4-11

J

JCL statement
AMBLIST service aid 15-2
SPZAP service aid 16-15
JES common coupling services
component trace 11-60
JES2 subsystem
component trace 11-71
job control language statement in IFCDIP00 14-4
JCL example 14-4
JOBNAMEP trace option
in GTF 10-20, 10-28
prompting 10-28
JPA parameter
dump option 5-9, 6-5

K

key-length-data format
SDWAVRA 14-25
keyboard A-1

L

library lookaside
See LLA 11-73
licensed documents xviii
link pack area
AMBLIST service aid 15-5
map 15-5
mapping using AMBLIST service aid 15-47
mapping with AMBLIST service aid 15-12
linkage stack
analysis for diagnosis 7-3
list a link pack area 15-5
list CSECT identification record 15-5
LIST service aid
control statement
LISTLOAD statement 16-13
description 15-1
planning 15-1
LISTIDR control statement
example 15-10, 15-12
format 15-5
LISTIDR control statement *(continued)*
in AMBLIST service aid 15-5, 15-10, 15-12
OUTPUT= 15-5
MODLIB parameter 15-5
parameter
DDN= 15-5
MEMBER= 15-5
TITLE= 15-5
LISTLOAD control statement
example 15-9, 15-10, 15-13
format 15-3
in AMBLIST service aid 15-3, 15-9, 15-10, 15-13
in LIST 16-13
parameter
ADATA= 15-3
DDN= 15-3
MEMBER= 15-3
OUTPUT= 15-3
RELOC= 15-3
TITLE= 15-3
to list the SSI 16-13
LISTLPA control statement
example 15-13
FLPA parameter 15-5
format 15-5
in AMBLIST service aid 15-5, 15-6, 15-13
MLPA parameter 15-6
PLPA parameter 15-6
LISTOBJ control statement
DDN parameter 15-4
example 15-7, 15-8, 15-10
format 15-4
in AMBLIST service aid 15-4, 15-7, 15-8, 15-10
parameter
MEMBER= 15-4
TITLE= 15-4
LLA (library lookaside)
component trace 11-73
LMI recovery record 14-8
LNKLIB parameter
of AMDSADMP macro 4-15
load module
AMBLIST service aid output 15-3
listing 15-3
listing current information using AMBLIST service aid 15-8
load module list
AMBLIST service aid output 15-9
locating the logrec buffer 14-18
locating the WTO buffer 14-18
locked processing mode
problem data 7-4
log stream
JCL specification 14-12
LOGDATA verb 14-18
logrec
buffer, recording control 14-18
format of buffer 14-19
formatting 14-18
how to print 14-11
recording control buffer 14-18

LOGREC Buffer
  format in a dump 2-44, 3-22
logrec data set
  error recording 14-6
    purpose 14-1
  header record
    format 14-8
    function 14-7
  initializing 14-2
  must be permanently mounted volume 14-4
  non-sharable data set 14-7
  purpose of error record 14-1
  reinitialization
    JCL example 14-3
  reinitializing 14-2
  reIPL
    JCL example 14-3
  size 14-2
  space allocation 14-3
    JCL example 14-3
    reallocating 14-3
  time stamp record 14-8
    format 14-8
  type of record 14-8
  type of record recorded 14-8
logrec data set - IFCDIP00 reallocation
  description 14-3
  JCL example 14-3
logrec data set - IFCDIP00 reinitialization
  description 14-4
  JCL example 14-4
logrec data set initialization 14-2, 14-3
Logrec data set service aid
  reasons for selection 1-5
logrec data set software record
  interpretation 14-19
  SDWA-type 14-19
    output 14-20
  symptom record 14-19
    output 14-26
logrec error record
  contents 14-7
  customize 14-28
  for diagnosis 14-1
logrec log stream
  defining 14-4
  error recording 14-6
    purpose 14-1
  example of creating a history data set 14-17
  example of producing an event history 14-18
  example of using system logger utility 14-5
  obtaining record 14-12
  purpose of error record 14-1
  type of record 14-8
  type of record recorded 14-8
logrec recording medium
  planning 14-2
LookAt message retrieval tool xviii
lost data records
  GTF trace record
    unformatted 10-79
lost event record
  in GTF trace 10-40
LPA parameter
  dump option 5-9, 6-5
LSQA dumped by stand-alone dump 4-5
LSQA parameter
  dump option 5-9, 6-5
LSR trace record
  formatted 10-54

M

machine check
  problem data 7-6
macro expansion messages
  in stand-alone dump 4-6
main storage dump
  description 4-1
  of stand-alone dump 4-5
map
  link pack area 15-5, 15-13
  nucleus 15-13
master trace 9-1, 9-5
  and hardcopy log 9-1
  customize 9-2
  dump output 9-4
  entry in trace table 9-6
  formatting 9-4
  header in trace table 9-5
  receive 9-3
  request 9-2
  start 9-2
  stop 9-2
MCH record 14-8
MCH trace entry
  in system trace 8-15
MCIC (machine check interrupt code)
  problem identification 7-6
MDR record 14-9
MEMBER= parameter
  LISTIDR control statement 15-5
  LISTLOAD control statement 15-3
  LISTOBJ control statement 15-4
membername
  for GTF 10-11
merge
  component and GTF trace 10-33
MERGE subcommand
  combining and formatting component trace data 11-24
message
  customize location 18-4
  customize processing 18-4
  from SPZAP 16-30
  produce 18-1
  receive 18-2
message display
  3480 device 4-49
  3490 device 4-49
  3590 device 4-49
  stand-alone dump program 4-49

message retrieval tool, LookAt xviii
messages 18-1
MIH record 14-9
MINASID parameter
  of AMDSADMP macro 4-14
MLPA parameter
  of LISTLPA control statement 15-6
MOBR trace entry
  in system trace 8-11
mode
  cross memory processing mode 7-4
  locked processing mode 7-4
  of processing 7-4
  physically disabled processing mode 7-4
  service request processing mode 7-4
  task processing mode 7-4
MODE trace entry
  in system trace 8-11
MODE= parameter
  in GTF 10-5
modified link pack area
  map 15-5
MODLIB parameter
  of AMDSADMP macro 4-15
  of LISTIDR control statement 15-5
module
  problem data 7-4
module summary
  AMBLIST output for module processed by binder 15-15, 15-48
  AMBLIST output for module processed by linkage editor 15-14, 15-47
MSCH trace entry
  in system trace 8-13
MSCH trace option
  in GTF 10-20
MSCH trace record
  formatted 10-55
MSG= parameter
  of AMDSADMP macro 4-14
multiple error events 2-44, 3-22

N

NAME control statement
  example 16-3, 16-4, 16-5, 16-6, 16-21
  in SPZAP 16-2, 16-3, 16-4, 16-5, 16-6, 16-21, 16-24
  parameter 16-24
NOALL parameter
  dump option 2-22, 3-8
NOALLPSA parameter
  dump option 2-22, 3-8
NODEFAULTS parameter
  dump option 2-22, 3-8
NOPROMPT parameter
  in GTF 10-6
NOPSA parameter
  dump option 2-22, 3-8
NOSQA parameter
  dump option 2-22, 3-8
NOSUM parameter
  dump option 2-22, 3-8
NOSYM parameter
  dump option 5-9
Notices B-1
notification
  of rapidly recurring dumps 17-4
NP parameter
  in GTF 10-6
NUC parameter
  dump option 5-9, 6-5
nucleus
  dump 2-30, 3-11, 5-15
  mapping using AMBLIST service aid 15-13
NUCLIB parameter
  of AMDSADMP macro 4-15

O

object module list
  obtain 15-8
object module or GOFF
  listing current information using AMBLIST service aid 15-6
OBR record 14-9
on EOD record 14-8
OPS (operations services component)
  component trace 11-92
option
  for dump content 2-16, 5-9
output 4-43, 4-44
  of AMBLIST service aid 15-13
  of SPZAP 16-29
output to DASD dump program
  example 4-17
OUTPUT= parameter
  LISTIDR control statement 15-5
  LISTLOAD control statement 15-3
  of AMDSADMP macro 4-13
overlay, storage
  in pattern recognition 7-1

P

pageable link pack area
  map 15-5
PARM option
  IGNIDRFULL 16-16
  of JCL EXEC statement
    for SPZAP service aid 16-16
parmlib library
  IEADMCxx member 2-14
  IEASLPxx member 2-13
PC trace entry
  in system trace 8-9
PCDATA parameter
  dump option 5-9, 6-5
PCI trace option
  in GTF 10-20
PCITAB parameter of GTF CCWP 10-25, 10-26

PGM trace entry
  in system trace 8-19
PGM trace record
  formatted 10-56
physically disabled processing mode
  problem data 7-4
PI trace option
  in GTF 10-21
PI trace record
  comprehensive
    unformatted 10-88
  formatted 10-56
  minimal
    unformatted 10-88
PIP trace option
  in GTF 10-21
PLPA parameter
  of LISTLPA control statement 15-6
PR trace entry
  in system trace 8-9
print
  ABEND dump 5-7
  SNAP dump 6-4
  stand-alone dump 4-48
  SVC dump 2-20
  SYSABEND dump 5-8
  Transaction dump 3-6
problem
  hardware-detected 5-21
  software-detected 5-20
problem data
  from dump 7-1
  from linkage stack 7-3
  from module 7-4
program
  system trace event 8-19
program check
  saved problem data 4-61
program event
  trace with GTF 10-1
program object
  AMBLIST service aid output 15-3
  listing 15-3
  listing current information using AMBLIST service aid 15-8
  using SPZAP to modify a CSECT
    example 16-8
program object module
  using SPZAP to modify a record
    example 16-12
PROMPT parameter
  of AMDSADMP macro 4-14, 4-18
prompting
  how to request 10-23
  in GTF 10-23
PSA dumped by stand-alone dump 4-5
PSA parameter
  dump option 2-22, 3-8
PSW parameter
  dump option 5-9, 6-5
PSWREGS data
  in dump 2-35
PT trace entry
  in system trace 8-9

Q

Q parameter
  dump option 5-9, 6-5

R

RCVY trace entry
  in system trace 8-20
RDE option 14-8
  on IPL record 14-8
real storage manager
  See RSM 11-105
reason code
  issued by stand-alone dump 4-37, 4-41
receive
  master trace 9-3
record
  error 14-1
  from logrec data set or logrec log stream 14-19
  GTF trace 10-34
  recording on logrec data set 14-8
  recording on logrec log stream 14-8
RECORD control statement
  in SPZAP 16-10
recordable extension
  in SDWA 14-25
recording control buffer 14-18
recording control buffer (RCB) 14-18
records on logrec data set 14-8
records on logrec log stream 14-8
recovery
  system trace event 8-20
recovery routine
  in dump customization 5-16, 5-18, 5-19
recovery work area
  problem data 7-4
registers
  format in a dump 2-47, 3-25
REGS parameter
  dump option 5-9, 6-6
reinitialize the logrec data set
  uncorrectable error 14-4
RELOC= parameter
  of LISTLOAD control statement 15-3
REP control statement
  example 16-3, 16-4, 16-5, 16-6, 16-21
  in SPZAP 16-2, 16-3, 16-4, 16-5, 16-6, 16-21, 16-25, 16-26
  variable 16-25, 16-26
resume subchannel data
  record 10-21
retention
  message
    by AMRF 18-5

retrieve information from logrec data set
using the EREP program 14-10
return code 4-30
from SPZAP 16-27
REUSEDS parameter
of AMDSADMP macro 4-14
RGN parameter
dump option 5-9
RNIO formatted trace record
formatted 10-57
RNIO trace option
in GTF 10-21
RR trace record
comprehensive
unformatted 10-88
minimal
unformatted 10-89
RRS (resource recovery services)
component trace 11-97
RSCH trace entry
in system trace 8-13
RSCH trace record
formatted 10-58
RSM (real storage manager)
component trace 11-105
RST trace entry
in system trace 8-15

S

S|I|SI parameter of GTF CCWP 10-25
SA parameter
dump option 5-9, 6-6
SA= parameter
in GTF 10-6
SADMP message
display example 4-49
MSADMP#U 4-49
NTRDY message 4-49
RSADMP# 4-49
RSADMP# U 4-49
SADMP# 4-49
status information
3480 device 4-49
SADMP= parameter
in GTF 10-6
SAH parameter
dump option 5-9, 6-6
scheduled SVC dump 2-39
SD= parameter
in GTF 10-6
SDSP trace record
formatted 10-44
SDUMP= parameter
in GTF 10-6
SDUMPX 4K SQA buffer
content 2-50
SDUMPX macro
to request SVC dump 2-12
SDWA (system diagnostic work area)
recordable extension 14-25
SDWAVRA (SDWA variable recording area)
key-length-data format 14-25
secondary extent
allocation 2-7, 2-8, 3-2, 3-3
calculation 2-8, 3-3
SPACE requirement 2-7, 3-2
security
problems with APPC/MVS component 11-35
self-dump
of stand-alone dump 4-41
sequential data set
extended
to hold large dumps 2-7
service aid selection
AMBLIST 1-2
DAE 1-2
IPCS 1-2
Logrec data set 1-2
SLIP 1-2
SPZAP 1-2
service aids and tools
descriptions 1-3
selection 1-1
service request processing mode
problem data 7-4
SETRP macro
for requesting ABEND dumps 5-6
in dump customization 5-16, 5-18, 5-19
SETSSI control statement
example 16-3, 16-4, 16-5, 16-6
in SPZAP 16-3, 16-4, 16-5, 16-6, 16-13, 16-26
parameter 16-26
shortcut keys A-1
SIGA trace entry
in system trace 8-13
SIO trace option
in GTF 10-21
SIOP trace option
in GTF 10-21
SLH record 14-9
SLIP command
SDUMPX 4K SQA buffer data 2-50
SLIP work area data 4-60
SLIP data
record 10-21
SLIP debug trace record
formatted 10-63
unformatted 10-92
SLIP operator command
control dump contents 2-29, 3-11
to request SVC dump 2-13
to suppress dumps 17-11
SLIP service aid
reasons for selection 1-6
SLIP standard trace record
formatted 10-62
unformatted 10-91
SLIP trace option
in GTF 10-21
SLIP user trace record
formatted 10-62, 10-63

SLIP user trace record *(continued)*
unformatted 10-91
SLIP work area
content 4-60
SNAP dump
contents 6-4
customization 6-4
introduction 6-1
reasons for selection 1-3
request 6-1
SNAP macro
for requesting dumps 6-3
SNAP or SNAPX macro
in dump customization 5-16, 5-17, 5-19
to request SNAP dump 6-3
software
error 14-1
software record 14-9, 14-19
detail edit report 14-20
interpretation 14-20
source index record
in GTF trace 10-40
source JCL
for component trace external writer 11-14
SPER trace entry
in system trace 8-19
SPI (service processor interface)
component trace 11-120
SPLS option
in dump customization 5-15, 5-16, 5-19
SPLS parameter
dump option 5-9, 6-6
SPZAP example
inspect and modify CSECT in HFS 16-7
SPZAP service aid
access
data record 16-12
accessing a load module 16-9
control statement
* 16-22
ABSDUMP 16-11, 16-18
ABSDUMPT 16-11, 16-18
BASE statement 16-9, 16-19, 16-21
CCHHR statement 16-10, 16-21
CHECKSUM statement 16-22
CONSOLE statement 16-22, 16-23
DUMP 16-3, 16-4, 16-5, 16-6, 16-21, 16-23
DUMPT 16-3, 16-4, 16-5, 16-6, 16-21, 16-23
IDRDATA statement 16-3, 16-4, 16-5, 16-6, 16-21, 16-24
NAME statement 16-2, 16-3, 16-4, 16-5, 16-6, 16-21, 16-24
RECORD statement 16-10
REP statement 16-2, 16-3, 16-4, 16-5, 16-6, 16-21, 16-25
rules for coding 16-17
SETSSI statement 16-3, 16-4, 16-5, 16-6, 16-13, 16-26
VERIFY statement 16-2, 16-3, 16-4, 16-5, 16-6, 16-21, 16-27
SPZAP service aid *(continued)*
data record
inspection 16-2, 16-10
modification 16-2, 16-10
description 16-1
dynamic invocation
example 16-29
macro form 16-28
example
running 16-3
JCL statement 16-15
load module
inspection 16-2
modification 16-2
monitoring SPZAP use 16-1
operational consideration 16-14
output 16-29
program object
inspection 16-2
modification 16-2
reasons for selection 1-6
return code 16-27
updating system status information 16-13
SQA buffer for SDUMPX
content 2-50
SQA dumped by stand-alone dump 4-5
SQA parameter
dump option 5-9, 6-6
SRB formatted trace record
in GTF trace 10-64
SRB trace entry
in system trace 8-12
SRM data
record 10-21
SRM trace option
in GTF 10-21
SRM trace record
comprehensive
unformatted 10-93
formatted 10-65
minimal
unformatted 10-93
SS trace entry
in system trace 8-15
SSAR trace entry
in system trace 8-9
SSCH trace entry
in system trace 8-13
SSCH trace option
in GTF 10-21
SSCH trace record
formatted 10-66
unformatted 10-93
SSCHP trace option
in GTF 10-21, 10-28
prompting 10-28
SSI (system status index)
field 16-13
flag byte 16-13, 16-14
SSRB trace entry
in system trace 8-12

SSRV trace entry
  in system trace 8-25
STAE trace record
  formatted 10-67
stand-alone dump 4-1
  analysis
    collecting initial data 4-50
    determining system state 4-51
    disabled loop 4-58
    disabled wait 4-57
    enabled loop 4-58
    enabled wait 4-53
    gathering external symptoms 4-50
    gathering IPCS symptoms 4-51
    problem data saved 4-60
  analyzing 4-50
  containing component trace records 11-12
  data space 4-21
  dump tailoring option, dumping additional storage 4-19
  reasons for selection 1-3
  service aid 4-1, 4-19
    3480 message 4-49
    3490 message 4-49
    3590 message 4-49
    AMDSADMP macro, coding 4-16
    assembly of the AMDSADMP macro 4-31
    central storage dump 4-5
    coding AMDSADMP macro, for high-speed dump 4-11
    copying dumps, DASD to DASD 4-45
    copying dumps, multiple devices to DASD 4-46
    copying dumps, tape to DASD 4-45
    creation 4-6
    data space 4-21
    DD statement 4-30
    device selection 4-6
    dumping 4-41
    dumping additional storage 4-17, 4-18
    dvolser 4-49
    error condition 4-6
    example 4-16
    generation, requesting additional storage 4-17
    initialization of residence volume 4-33
    initialization of resident volume 4-6
    instruction address trace 4-5
    IPCS LIST subcommand, self-dump 4-41
    macro message 4-30
    macro parameter 4-11, 4-12, 4-13, 4-14, 4-15, 4-18, 4-32
    main storage dump 4-5
    mapping, nucleus 4-5
    message output 4-6
    nucleus 4-5
    one-step generation 4-6, 4-28, 4-31
    output 4-43
    printing dumps, using IPCS 4-48
    reason code 4-37, 4-41
    reinitializing 4-41
    residence volume initialization 4-6
    restart 4-40
stand-alone dump *(continued)*
  service aid *(continued)*
    restarting stand-alone dump 4-40
    return code 4-30
    running 4-18, 4-36, 4-37, 4-41
    running stand-alone dump in a sysplex 4-41
    running the dump program 4-37, 4-40, 4-41
    sample JCL 4-28, 4-32
    self-dump 4-41
    stage-two generation 4-31, 4-33
    storage dump 4-41
    system restart 4-40
    two-stage generation 4-6
    unformatted output 4-43, 4-44
    viewing dumps, using IPCS 4-47
    virtual storage dump 4-5
    wait state 4-37, 4-41
    wait-reason code 4-39
    wait-reason code, processing completion 4-40
  specification
    of address range 4-19
    of subpool 4-19
stand-alone dump example
  analyzing a disabled wait 4-57
  determining if TCB in normal wait 4-57
  determining ready work 4-56
  determining resource contention 4-54
  determining system activity 4-54
  determining the module 4-53
  determining the system state 4-52
  gather symptom data 4-51
  obtaining real storage data 4-55
  obtaining storage data 4-55
  of using stand-alone dump 4-16
  reading the instruction address trace 4-59
  reading the PSW 4-53
  recognizing an enabled loop 4-58
stand-along dump
  service aid
    output 4-44
START command 10-16
  for GTF 10-11
start subchannel data
  record 10-21
status
  of master tracing 9-2
  of SYS1.DUMPxx data set 2-16
STATUS subcommand
  FAILDATA report 2-40, 3-17
  REGISTERS report 2-47, 3-25
  SYSTEM report 2-39, 3-17
  WORKSHEET report 2-38, 3-15
STOP command 10-16, 10-17
storage overlay
  determination
    damaged area 7-1
  in pattern recognition 7-1
STORE STATUS
  stand-alone dump 4-37
STORE STATUS command 4-5, 4-37, 4-41

striping
  use for dumps 2-9
sublevel
  component trace 11-1
  component traces 11-3
  verifying sublevel traces 11-21
SUBPLST option
  in dump customization 5-15, 5-16, 5-19
subpools dumped by stand-alone dump 4-5
SUBSYS trace record
  formatted 10-50
SUBTASKS parameter
  dump option 2-25, 5-9, 6-6
  in macro parameter list 5-15
SUM parameter
  dump option 5-9
SUMDUMP output 2-32, 2-33, 3-13, 3-14
SUMLIST address range
  in dump 2-34, 2-35, 3-14
SUMLSTA address range
  in dump 2-34, 2-35, 3-14
summary dump 2-32, 3-13
  disabled 2-26
  enabled 2-26
  in ABEND dump 5-13
  in SVC dump 2-26
  in Transaction dump 3-11
  suspend 2-26
SUMMARY subcommand
  TCBERROR report 2-43, 3-20
summary SVC dump 2-1
supervisor
  system trace event 8-24
suppression
  generating a suppressed dump 17-9
  of dumps 17-1
  of dumps by abend code 17-11
  of dumps by ABEND macro 17-12
  of dumps by DAE 17-1
  of dumps by installation exit routines 17-12
  of dumps by RACF 17-12
  of dumps by SLIP trap 17-11
  of messages 18-4
  of rapidly recurring dumps 17-3
  of symptom dumps 18-5
  reasons dumps are suppressed 17-12
SUSP trace entry
  in system trace 8-17
suspend
  summary dump 2-26
suspend lock
  system trace event 8-17
SUSPEND parameter
  to control summary dump 2-26
SVC D
  example in SYSTRACE report 2-47, 3-25
SVC dump
  analyze using IPCS 2-36
  asynchronous 2-1
  automatically allocated data set 2-2
  clearing 2-20
SVC dump *(continued)*
  containing component trace records 11-12
  contents 2-21
  copying 2-20
  customization 2-21
  data set 2-15
  debugging hint 2-32
  displaying options 2-16
  DUMPDS command 2-15
  introduction 2-1
  planning dump data set 2-2
  pre-allocated data set 2-7
  printing 2-20
  reasons for selection 1-4
  request 2-11
  scheduled 2-1
  SUMDUMP output for branch-entry SDUMPX 2-34
  SUMDUMP output for SVC dump 2-33
  summary dump 2-1
  summary dump contents 2-26
  suppressing 17-1
  symptom dump 18-3
  synchronous 2-1
  viewing 2-20
SVC dump example
  DISPLAY DUMP,ERRDATA command 2-19
  IPCS VERBX SUMDUMP command 2-33
SVC interruption
  record 10-22
SVC trace entry
  in system trace 8-24
SVC trace option
  in GTF 10-22
SVC trace record
  comprehensive
    unformatted 10-94
  formatted 10-69
  minimal
    unformatted 10-95
SVCE trace entry
  in system trace 8-24
SVCP trace option
  in GTF 10-22, 10-28
  prompting 10-28
SVCR trace entry
  in system trace 8-24
SWA parameter
  dump option 5-9, 6-6
symptom
  display in SVC dumps 2-18
symptom dump
  receive 18-3
  suppress 18-5
symptom record 14-19
  detail edit report 14-26
  interpretation 14-26
symptom string
  generating a suppressed dump 17-9
  viewing in DAE data set 17-8
SYMREC macro 14-26
synchronous SVC dump 2-39

synchronous Transaction dump 3-17
SYS trace option
  combining certain trace options 10-23
  in GTF 10-22, 10-23
SYS1.DUMPxx data set
  availability 2-16
  control 2-10
  determine contents 2-17
  determine status 2-16
  displaying information 18-3
  to receive SVC dump 2-12
    specify 2-10
SYS1.LPALIB library 16-15
SYS1.MACLIB
  AMDSADMP macro
    assembly 4-32
SYSABEND
  analysis approach 5-1
SYSABEND DD statement 16-17
SYSABEND dump
  analysis 5-20
  customization 5-16, 5-17
  symptom dump 18-3
SYSAPPC component trace
  CTnAPPxx parmlib member 11-26
  FMH-5 trace data 11-35
  format options 11-29
  FULL report 11-34
  SHORT report 11-33
  SUMMARY report 11-33
  TRACE command 11-26
  trace request options 11-26
SYSDLF component trace
  FULL report 11-40
SYSDSOM component trace
  FULL report 11-42
SYSGRS component trace
  CTnGRSxx parmlib member 11-44
  SHORT report 11-48
  TALLY report 11-48
  TRACE command 11-44
  trace request options 11-44
SYSIEFAL component trace
  CTIIEFxx parmlib member 11-49
  FULL report 11-53
  SHORT report 11-53
  TRACE command 11-50
  trace request options 11-49
SYSIN DD statement 16-4, 16-17, 16-21
  in stand-alone dump 4-30
  used in AMBLIST service aid 15-2
  used in SPZAP 16-23
SYSIOS component trace
  CTnIOSxx parmlib member 11-55
  SHORT report 11-58
  TRACE command 11-56
  trace request options 11-55
SYSJES component trace
  CTnJESxx parmlib member 11-62
  format options 11-65
  request sublevels 11-61
SYSJES component trace *(continued)*
  requesting 11-62
  TRACE command 11-63
  verifying 11-21
SYSjes2 component trace
  format options 11-71
  request sublevels 11-71
  requesting 11-71
SYSLIB DD statement 16-2, 16-3, 16-4, 16-5, 16-11, 16-15, 16-16, 16-21
  in stand-alone dump 4-31
  used in SPZAP 16-23
SYSLOGR component trace
  CTnLOGxx parmlib member 11-77
  output 11-81
  trace request options 11-78
SYSM trace option
  in GTF 10-22
SYSMDUMP
  analysis approach 5-1
SYSMDUMP dump
  customization 5-17, 5-18
  preallocated data set 5-4
  suppressing 17-1
  symptom dump 18-3
  to VIO data set 5-4
SYSOMVS component trace
  CTnBPXxx parmlib member 11-82
  format options 11-84
  FULL report 11-86
  SUMMARY report 11-91
  TRACE command 11-82
  trace request options 11-83
SYSOPS component trace
  CTnOPSxx parmlib member 11-93
  format options 11-95
  TRACE command 11-93
  trace options 11-94
  trace request options 11-93
SYSOUT data set
  to receive ABEND dump 5-7
SYSP trace option
  in GTF 10-22
sysplex
  component trace data sets 11-14
  component tracing in systems 11-18
SYSPRINT DD
  used in AMBLIST service aid 15-2
SYSPRINT DD statement 16-11, 16-16
  in stand-alone dump 4-30
SYSPUNCH DD statement
  in stand-alone dump 4-32
SYSRRS component trace
  CTnRRSxx parmlib member 11-98
  FULL report 11-104
  SHORT report 11-103
  SUMMARY report 11-104
  TALLY report 11-105
  TRACE command 11-98
  trace request options 11-98

SYSRSM component trace
  CTnRSMxx parmlib member 11-106
  FULL report 11-119
  TRACE command 11-107
  trace request options 11-106
system control (CC-012) frame 4-37
system data records
  GTF trace record
    unformatted 10-81
system event
  trace with GTF 10-1
System logger component
  component trace 11-74
system mode
  in STATUS FAILDATA report 2-41, 3-18
system resource manager data
  record 10-21
system restart
  for stand-alone dump 4-40
system service
  system trace event 8-25
system status index
  function 16-13
system trace 8-1
  ACR trace entry 8-6
  addressing mode trace entries 8-11
  altering options 8-7
  ALTR trace entry 8-7
  branch trace entries 8-8
  BSG trace entry 8-9
  CALL trace entry 8-15
  CLKC trace entry 8-15
  CSCH trace, entry 8-13
  customizing 8-1
  DSP trace entry 8-12
  EMS trace entry 8-15
  entries 8-3
  entry identifiers 8-4
  example in a dump 8-3
  EXT trace entry 8-15
  format in a dump 2-47, 3-25
  formatting output 8-3
  HSCH trace entry 8-13
  I/O trace entry 8-15
  increasing the size of trace table 8-1
  MCH trace entry 8-15
  MOBR trace entry 8-11
  MODE trace entry 8-11
  MSCH trace entry 8-13
  PC trace entry 8-9
  PGM trace entry 8-19
  PR trace entry 8-9
  PT trace entry 8-9
  RCVY trace entry 8-20
  reading output 8-3
  receiving output in a dump 8-2
  RSCH trace entry 8-13
  RST trace entry 8-15
  SPER trace entry 8-19
  SRB trace entry 8-12
  SS trace entry 8-15
system trace *(continued)*
  SSAR trace entry 8-9
  SSCH trace entry 8-13
  SSRB trace entry 8-12
  SSRV trace entry 8-25
  SUSP trace entry 8-17
  SVC trace entry 8-24
  SVCE trace entry 8-24
  SVCR trace entry 8-24
  TIME trace entry 8-28
  tracing branch instructions 8-2
  USRn trace entry 8-29
  WAIT trace entry 8-12
SYSTEM trace record
  formatted 10-50
system trace table
  increasing 8-1
SYSTRACE subcommand
  dump output 8-3
SYSUDUMP
  analysis approach 5-1
SYSUDUMP dump
  analysis 5-20
  customization 5-19, 5-20
  symptom dump 18-3
SYSUT= parameter
  of AMDSADMP macro 4-12
SYSUT2 DD statement 4-45
SYSVLF component trace
  FULL report 11-123
  trace request options 11-122
SYSWLM component trace
  format options 11-126
  FULL report 11-126
  TRACE command 11-125
  trace request options 11-125
SYSXCF component trace
  CTnXCFxx parmlib member 11-128
  format options 11-130
  FULL report 11-131
  TRACE command 11-129
  trace request options 11-128
SYSXES component trace
  CTnXESxx parmlib member 11-133
  format options 11-135
  SHORT report 11-136
  TRACE command 11-134
  trace request options 11-133, 11-134
  verifying 11-21

T

task processing mode
  problem data 7-4
time
  display in dump 2-17
  system trace event 8-28
time of dump 2-39, 2-40, 3-17
time stamp record
  in GTF trace 10-39

time stamp record *(continued)*
logrec data set
format 14-8
how recorded 14-8
updated at user-specified interval 14-8
TIME trace entry
in system trace 8-28
TIME= parameter
in GTF 10-7
title
display in dump 2-17
TITLE= parameter
LISTIDR control statement 15-5
LISTLOAD control statement 15-3
LISTOBJ control statement 15-4
tools and service aids
overview 1-3
selection 1-1
trace a functional recovery routine operation 10-21
trace a modify subchannel operation 10-20
trace a program interruption 10-21
trace a SLIP trap 10-21
trace an I/O interruption 10-19
TRACE command
for SYSAPPC component trace 11-26
for SYSGRS component trace 11-44
for SYSIEFAL component trace 11-50
for SYSIOS component trace 11-56
for SYSJES component trace 11-63
for SYSOMVS component trace 11-82
for SYSOPS component trace 11-93
for SYSRRS component trace 11-98
for SYSRSM component trace 11-107
for SYSWLM component trace 11-125
for SYSXCF component trace 11-129
for SYSXES component trace 11-134
trace data set
for component trace 11-13
TRACE operator command
determining master trace status 9-2
trace selection
component trace 1-2
GFS trace 1-2
GTF trace 1-2
master trace 1-2
system trace 1-2
trace table
for master trace 9-5
trace table in storage 9-5
trace VTAM network activity 10-21
traces
description 1-4, 1-5
Transaction dump
analyze using IPCS 3-14
asynchronous 3-1
automatically allocated data set 3-3
clearing 3-6
contents 3-7
copying 3-6
customization 3-7
debugging hint 3-13
Transaction dump *(continued)*
planning dump data set 3-2
pre-allocated data set 3-2
printing 3-6
request 3-5
SUMDUMP output for Transaction dump 3-14
summary dump contents 3-11
synchronous 3-1
viewing 3-6
Transaction Dump
introduction 3-1
transaction trace
SYSTTRC for transaction trace 11-121
Transaction Trace
introduction 12-1
TRC trace option
in GTF 10-22
TRT parameter
dump option 5-9, 6-6
TTRC (transaction trace)
transaction trace 11-121
two-stage generation
migration 4-33
overriding 4-35
type of record
ANR record 14-8
CRW record 14-8
DASD-SIM recovery record 14-8
DDR record 14-8
EOD record 14-8
ETR recovery record 14-8
IOS recovery record 14-8
IPL record 14-8
LMI recovery record 14-8
MCH record 14-8
MDR record 14-9
MIH record 14-9
OBR record 14-9
SLH record 14-9
software record 14-9

U

ULABEL= parameter
of AMDSADMP macro 4-12
unformatted dump 4-1
SVC dump 2-20
SYSMDUMP dump 5-8
Transaction dump 3-6
unformatted dump program
example 4-16, 4-17
unformatted GTF records
control records 10-77
lost data records 10-79
system data records 10-81
user trace records 10-80
unformatted output
of GTF 10-77
user
system trace event 8-29

user data records
GTF trace record 10-80
unformatted 10-80
user trace data
record 10-23
USR trace option
in GTF 10-23
USR trace record
formatted 10-70
USRn trace entry
in system trace 8-29
USRP trace option
in GTF 10-23, 10-28
prompting 10-28

V

VERBEXIT MTRACE subcommand
dump output 9-4
to format master trace 9-3
VERBEXIT subcommand
LOGDATA report 2-44, 3-22
TRACE report 2-47, 3-25
verify
component trace 11-21
external writer 11-21
VERIFY control statement
example 16-3, 16-4, 16-5, 16-6, 16-21
in SPZAP 16-2, 16-3, 16-4, 16-5, 16-6, 16-21, 16-27
parameter 16-27
view
ABEND dump 5-7
SVC dump 2-20
Transaction dump 3-6
viewing
component trace data 11-23
virtual lookaside facility
See VLF 11-121
virtual storage dump
description 4-1
of stand-alone dump 4-5
VLF (virtual lookaside facility)
component trace 11-121
VOLSER= parameter
of AMDSADMP macro 4-12
VSAM object
access 16-17
VTAM network activity
record 10-21

W

wait state code
issued by stand-alone dump 4-37, 4-41
WAIT trace entry
in system trace 8-12
wait-reason code
issued by stand-alone dump 4-39
unload a tape 4-40
WLM (workload manager)
component trace 11-125
wrap
of component trace data sets 11-15
WRAP parameter
TRACE CT command 11-15
WTO recording control buffer location 14-18

X

XCF (cross-system coupling facility)
component trace 11-127
xsd listing
AMBLIST output for LISTOBJ with XSD record. 15-19

Z

z/OS UNIX
component trace 11-81
zeroed page dump suppression 4-5

Readers' Comments — We'd Like to Hear from You

z/OS
MVS Diagnosis:
Tools and Service Aids

Publication No. GA22-7589-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
GA22-7589-03

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

GA22-7589-03

Cut or Fold
Along Line

Program Number: 5694-A01, 5655-G52

Printed in U.S.A.

GA22-7589-03